PINESCRIPT V5 DOCUMENTATION

Search

**Variables**

[bar\_index](https://www.tradingview.com/pine-script-reference/v6/#var_bar_index)

[barstate.isconfirmed](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isconfirmed)

[barstate.isfirst](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isfirst)

[barstate.ishistory](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.ishistory)

[barstate.islast](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islast)

[barstate.islastconfirmedhistory](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islastconfirmedhistory)

[barstate.isnew](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isnew)

[barstate.isrealtime](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isrealtime)

[box.all](https://www.tradingview.com/pine-script-reference/v6/#var_box.all)

[chart.bg\_color](https://www.tradingview.com/pine-script-reference/v6/#var_chart.bg_color)

[chart.fg\_color](https://www.tradingview.com/pine-script-reference/v6/#var_chart.fg_color)

[chart.is\_heikinashi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_heikinashi)

[chart.is\_kagi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_kagi)

[chart.is\_linebreak](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_linebreak)

[chart.is\_pnf](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_pnf)

[chart.is\_range](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_range)

[chart.is\_renko](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_renko)

[chart.is\_standard](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_standard)

[chart.left\_visible\_bar\_time](https://www.tradingview.com/pine-script-reference/v6/#var_chart.left_visible_bar_time)

[chart.right\_visible\_bar\_time](https://www.tradingview.com/pine-script-reference/v6/#var_chart.right_visible_bar_time)

[close](https://www.tradingview.com/pine-script-reference/v6/#var_close)

[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)

[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)

[dividends.future\_amount](https://www.tradingview.com/pine-script-reference/v6/#var_dividends.future_amount)

[dividends.future\_ex\_date](https://www.tradingview.com/pine-script-reference/v6/#var_dividends.future_ex_date)

[dividends.future\_pay\_date](https://www.tradingview.com/pine-script-reference/v6/#var_dividends.future_pay_date)

[earnings.future\_eps](https://www.tradingview.com/pine-script-reference/v6/#var_earnings.future_eps)

[earnings.future\_period\_end\_time](https://www.tradingview.com/pine-script-reference/v6/#var_earnings.future_period_end_time)

[earnings.future\_revenue](https://www.tradingview.com/pine-script-reference/v6/#var_earnings.future_revenue)

[earnings.future\_time](https://www.tradingview.com/pine-script-reference/v6/#var_earnings.future_time)

[high](https://www.tradingview.com/pine-script-reference/v6/#var_high)

[hl2](https://www.tradingview.com/pine-script-reference/v6/#var_hl2)

[hlc3](https://www.tradingview.com/pine-script-reference/v6/#var_hlc3)

[hlcc4](https://www.tradingview.com/pine-script-reference/v6/#var_hlcc4)

[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)

[label.all](https://www.tradingview.com/pine-script-reference/v6/#var_label.all)

[last\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#var_last_bar_index)

[last\_bar\_time](https://www.tradingview.com/pine-script-reference/v6/#var_last_bar_time)

[line.all](https://www.tradingview.com/pine-script-reference/v6/#var_line.all)

[linefill.all](https://www.tradingview.com/pine-script-reference/v6/#var_linefill.all)

[low](https://www.tradingview.com/pine-script-reference/v6/#var_low)

[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)

[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)

[na](https://www.tradingview.com/pine-script-reference/v6/#var_na)

[ohlc4](https://www.tradingview.com/pine-script-reference/v6/#var_ohlc4)

[open](https://www.tradingview.com/pine-script-reference/v6/#var_open)

[polyline.all](https://www.tradingview.com/pine-script-reference/v6/#var_polyline.all)

[second](https://www.tradingview.com/pine-script-reference/v6/#var_second)

[session.isfirstbar](https://www.tradingview.com/pine-script-reference/v6/#var_session.isfirstbar)

[session.isfirstbar\_regular](https://www.tradingview.com/pine-script-reference/v6/#var_session.isfirstbar_regular)

[session.islastbar](https://www.tradingview.com/pine-script-reference/v6/#var_session.islastbar)

[session.islastbar\_regular](https://www.tradingview.com/pine-script-reference/v6/#var_session.islastbar_regular)

[session.ismarket](https://www.tradingview.com/pine-script-reference/v6/#var_session.ismarket)

[session.ispostmarket](https://www.tradingview.com/pine-script-reference/v6/#var_session.ispostmarket)

[session.ispremarket](https://www.tradingview.com/pine-script-reference/v6/#var_session.ispremarket)

[strategy.account\_currency](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency)

[strategy.avg\_losing\_trade](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.avg_losing_trade)

[strategy.avg\_losing\_trade\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.avg_losing_trade_percent)

[strategy.avg\_trade](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.avg_trade)

[strategy.avg\_trade\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.avg_trade_percent)

[strategy.avg\_winning\_trade](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.avg_winning_trade)

[strategy.avg\_winning\_trade\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.avg_winning_trade_percent)

[strategy.closedtrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.closedtrades)

[strategy.closedtrades.first\_index](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.closedtrades.first_index)

[strategy.equity](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.equity)

[strategy.eventrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.eventrades)

[strategy.grossloss](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.grossloss)

[strategy.grossloss\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.grossloss_percent)

[strategy.grossprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.grossprofit)

[strategy.grossprofit\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.grossprofit_percent)

[strategy.initial\_capital](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.initial_capital)

[strategy.losstrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.losstrades)

[strategy.margin\_liquidation\_price](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.margin_liquidation_price)

[strategy.max\_contracts\_held\_all](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_contracts_held_all)

[strategy.max\_contracts\_held\_long](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_contracts_held_long)

[strategy.max\_contracts\_held\_short](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_contracts_held_short)

[strategy.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_drawdown)

[strategy.max\_drawdown\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_drawdown_percent)

[strategy.max\_runup](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_runup)

[strategy.max\_runup\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_runup_percent)

[strategy.netprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit)

[strategy.netprofit\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit_percent)

[strategy.openprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.openprofit)

[strategy.openprofit\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.openprofit_percent)

[strategy.opentrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.opentrades)

[strategy.opentrades.capital\_held](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.opentrades.capital_held)

[strategy.position\_avg\_price](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_avg_price)

[strategy.position\_entry\_name](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_entry_name)

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)

[strategy.wintrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.wintrades)

[syminfo.basecurrency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.basecurrency)

[syminfo.country](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.country)

[syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency)

[syminfo.description](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.description)

[syminfo.employees](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.employees)

[syminfo.expiration\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.expiration_date)

[syminfo.industry](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.industry)

[syminfo.main\_tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.main_tickerid)

[syminfo.mincontract](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mincontract)

[syminfo.minmove](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.minmove)

[syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mintick)

[syminfo.pointvalue](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.pointvalue)

[syminfo.prefix](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.prefix)

[syminfo.pricescale](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.pricescale)

[syminfo.recommendations\_buy](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy)

[syminfo.recommendations\_buy\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy_strong)

[syminfo.recommendations\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_date)

[syminfo.recommendations\_hold](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_hold)

[syminfo.recommendations\_sell](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell)

[syminfo.recommendations\_sell\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell_strong)

[syminfo.recommendations\_total](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_total)

[syminfo.root](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.root)

[syminfo.sector](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.sector)

[syminfo.session](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.session)

[syminfo.shareholders](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.shareholders)

[syminfo.shares\_outstanding\_float](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.shares_outstanding_float)

[syminfo.shares\_outstanding\_total](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.shares_outstanding_total)

[syminfo.target\_price\_average](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_average)

[syminfo.target\_price\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_date)

[syminfo.target\_price\_estimates](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_estimates)

[syminfo.target\_price\_high](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_high)

[syminfo.target\_price\_low](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_low)

[syminfo.target\_price\_median](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_median)

[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)

[syminfo.timezone](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.timezone)

[syminfo.type](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.type)

[syminfo.volumetype](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.volumetype)

[ta.accdist](https://www.tradingview.com/pine-script-reference/v6/#var_ta.accdist)

[ta.iii](https://www.tradingview.com/pine-script-reference/v6/#var_ta.iii)

[ta.nvi](https://www.tradingview.com/pine-script-reference/v6/#var_ta.nvi)

[ta.obv](https://www.tradingview.com/pine-script-reference/v6/#var_ta.obv)

[ta.pvi](https://www.tradingview.com/pine-script-reference/v6/#var_ta.pvi)

[ta.pvt](https://www.tradingview.com/pine-script-reference/v6/#var_ta.pvt)

[ta.tr](https://www.tradingview.com/pine-script-reference/v6/#var_ta.tr)

[ta.vwap](https://www.tradingview.com/pine-script-reference/v6/#var_ta.vwap)

[ta.wad](https://www.tradingview.com/pine-script-reference/v6/#var_ta.wad)

[ta.wvad](https://www.tradingview.com/pine-script-reference/v6/#var_ta.wvad)

[table.all](https://www.tradingview.com/pine-script-reference/v6/#var_table.all)

[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)

[time\_close](https://www.tradingview.com/pine-script-reference/v6/#var_time_close)

[time\_tradingday](https://www.tradingview.com/pine-script-reference/v6/#var_time_tradingday)

[timeframe.isdaily](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdaily)

[timeframe.isdwm](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdwm)

[timeframe.isintraday](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isintraday)

[timeframe.isminutes](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isminutes)

[timeframe.ismonthly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.ismonthly)

[timeframe.isseconds](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isseconds)

[timeframe.isticks](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isticks)

[timeframe.isweekly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isweekly)

[timeframe.main\_period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.main_period)

[timeframe.multiplier](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.multiplier)

[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period)

[timenow](https://www.tradingview.com/pine-script-reference/v6/#var_timenow)

[volume](https://www.tradingview.com/pine-script-reference/v6/#var_volume)

[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)

[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)

**Constants**

**Functions**

**Keywords**

**Types**

**Operators**

**Annotations**

Version 3 Version 4 Version 5 Version 6

# **Pine Script® language reference manual**

## **Variables**

**bar\_index**

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

TYPE

series int

EXAMPLE

//**@version=**6

indicator("bar\_index")

plot(bar\_index)

plot(bar\_index > 5000 ? close : 0)

REMARKS

Note that **bar\_index** has replaced **n** variable in version 4.

Note that bar indexing starts from 0 on the first historical bar.

Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[last\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#var_last_bar_index)[barstate.isfirst](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isfirst)[barstate.islast](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islast)[barstate.isrealtime](https://www.tradingview.com/pine-script-reference/v6/#var_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.

It is NOT recommended to use [barstate.isconfirmed](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isconfirmed) in [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security) expression. Its value requested from [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security) is unpredictable.

Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[barstate.isfirst](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isfirst)[barstate.islast](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islast)[barstate.ishistory](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.ishistory)[barstate.isrealtime](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isrealtime)[barstate.isnew](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isnew)[barstate.islastconfirmedhistory](https://www.tradingview.com/pine-script-reference/v6/#var_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.

Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[barstate.islast](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islast)[barstate.ishistory](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.ishistory)[barstate.isrealtime](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isrealtime)[barstate.isnew](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isnew)[barstate.isconfirmed](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isconfirmed)[barstate.islastconfirmedhistory](https://www.tradingview.com/pine-script-reference/v6/#var_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.

Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[barstate.isfirst](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isfirst)[barstate.islast](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islast)[barstate.isrealtime](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isrealtime)[barstate.isnew](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isnew)[barstate.isconfirmed](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isconfirmed)[barstate.islastconfirmedhistory](https://www.tradingview.com/pine-script-reference/v6/#var_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.

Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[barstate.isfirst](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isfirst)[barstate.ishistory](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.ishistory)[barstate.isrealtime](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isrealtime)[barstate.isnew](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isnew)[barstate.isconfirmed](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isconfirmed)[barstate.islastconfirmedhistory](https://www.tradingview.com/pine-script-reference/v6/#var_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.

Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[barstate.isfirst](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isfirst)[barstate.islast](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islast)[barstate.ishistory](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.ishistory)[barstate.isrealtime](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isrealtime)[barstate.isnew](https://www.tradingview.com/pine-script-reference/v6/#var_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.

Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[barstate.isfirst](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isfirst)[barstate.islast](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islast)[barstate.ishistory](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.ishistory)[barstate.isrealtime](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isrealtime)[barstate.isconfirmed](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isconfirmed)[barstate.islastconfirmedhistory](https://www.tradingview.com/pine-script-reference/v6/#var_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.

Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[barstate.isfirst](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isfirst)[barstate.islast](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islast)[barstate.ishistory](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.ishistory)[barstate.isnew](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isnew)[barstate.isconfirmed](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.isconfirmed)[barstate.islastconfirmedhistory](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islastconfirmedhistory)

**box.all**

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

TYPE

array<box>

EXAMPLE

//**@version=**6

indicator("box.all")

//delete all boxes

box.new(time, open, time + 60 \* 60 \* 24, close, xloc=xloc.bar\_time, border\_style=line.style\_dashed)

a\_allBoxes = box.all

if array.size(a\_allBoxes) > 0

for i = 0 to array.size(a\_allBoxes) - 1

box.delete(array.get(a\_allBoxes, i))

REMARKS

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

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[line.all](https://www.tradingview.com/pine-script-reference/v6/#var_line.all)[label.all](https://www.tradingview.com/pine-script-reference/v6/#var_label.all)[table.all](https://www.tradingview.com/pine-script-reference/v6/#var_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

[chart.fg\_color](https://www.tradingview.com/pine-script-reference/v6/#var_chart.fg_color)

**chart.fg\_color**

Returns a color providing optimal contrast with [chart.bg\_color](https://www.tradingview.com/pine-script-reference/v6/#var_chart.bg_color).

TYPE

input color

SEE ALSO

[chart.bg\_color](https://www.tradingview.com/pine-script-reference/v6/#var_chart.bg_color)

**chart.is\_heikinashi**

TYPE

simple bool

RETURNS

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the chart type is Heikin Ashi, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SEE ALSO

[chart.is\_renko](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_renko)[chart.is\_linebreak](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_linebreak)[chart.is\_kagi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_kagi)[chart.is\_pnf](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_pnf)[chart.is\_range](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_range)

**chart.is\_kagi**

TYPE

simple bool

RETURNS

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the chart type is Kagi, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SEE ALSO

[chart.is\_renko](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_renko)[chart.is\_linebreak](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_linebreak)[chart.is\_heikinashi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_heikinashi)[chart.is\_pnf](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_pnf)[chart.is\_range](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_range)

**chart.is\_linebreak**

TYPE

simple bool

RETURNS

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the chart type is Line break, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SEE ALSO

[chart.is\_renko](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_renko)[chart.is\_heikinashi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_heikinashi)[chart.is\_kagi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_kagi)[chart.is\_pnf](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_pnf)[chart.is\_range](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_range)

**chart.is\_pnf**

TYPE

simple bool

RETURNS

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the chart type is Point & figure, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SEE ALSO

[chart.is\_renko](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_renko)[chart.is\_linebreak](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_linebreak)[chart.is\_kagi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_kagi)[chart.is\_heikinashi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_heikinashi)[chart.is\_range](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_range)

**chart.is\_range**

TYPE

simple bool

RETURNS

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the chart type is Range, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SEE ALSO

[chart.is\_renko](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_renko)[chart.is\_linebreak](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_linebreak)[chart.is\_kagi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_kagi)[chart.is\_pnf](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_pnf)[chart.is\_heikinashi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_heikinashi)

**chart.is\_renko**

TYPE

simple bool

RETURNS

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the chart type is Renko, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SEE ALSO

[chart.is\_heikinashi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_heikinashi)[chart.is\_linebreak](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_linebreak)[chart.is\_kagi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_kagi)[chart.is\_pnf](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_pnf)[chart.is\_range](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_range)

**chart.is\_standard**

TYPE

simple bool

RETURNS

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the chart type is not one of the following: Renko, Kagi, Line break, Point & figure, Range, Heikin Ashi; [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SEE ALSO

[chart.is\_renko](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_renko)[chart.is\_linebreak](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_linebreak)[chart.is\_kagi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_kagi)[chart.is\_pnf](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_pnf)[chart.is\_range](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_range)[chart.is\_heikinashi](https://www.tradingview.com/pine-script-reference/v6/#var_chart.is_heikinashi)

**chart.left\_visible\_bar\_time**

The [time](https://www.tradingview.com/pine-script-reference/v6/#var_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.

Alerts created on a script that includes this variable will only use the value assigned to the variable at the moment of the alert's creation, regardless of whether the value changes afterward, which may lead to repainting.

SEE ALSO

[chart.right\_visible\_bar\_time](https://www.tradingview.com/pine-script-reference/v6/#var_chart.right_visible_bar_time)

**chart.right\_visible\_bar\_time**

The [time](https://www.tradingview.com/pine-script-reference/v6/#var_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.

Alerts created on a script that includes this variable will only use the value assigned to the variable at the moment of the alert's creation, regardless of whether the value changes afterward, which may lead to repainting.

SEE ALSO

[chart.left\_visible\_bar\_time](https://www.tradingview.com/pine-script-reference/v6/#var_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

[open](https://www.tradingview.com/pine-script-reference/v6/#var_open)[high](https://www.tradingview.com/pine-script-reference/v6/#var_high)[low](https://www.tradingview.com/pine-script-reference/v6/#var_low)[volume](https://www.tradingview.com/pine-script-reference/v6/#var_volume)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[hl2](https://www.tradingview.com/pine-script-reference/v6/#var_hl2)[hlc3](https://www.tradingview.com/pine-script-reference/v6/#var_hlc3)[hlcc4](https://www.tradingview.com/pine-script-reference/v6/#var_hlcc4)[ohlc4](https://www.tradingview.com/pine-script-reference/v6/#var_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

[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofmonth)[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#var_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.

You can use [dayofweek.sunday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.sunday), [dayofweek.monday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.monday), [dayofweek.tuesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.tuesday), [dayofweek.wednesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.wednesday), [dayofweek.thursday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.thursday), [dayofweek.friday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.friday) and [dayofweek.saturday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.saturday) variables for comparisons.

SEE ALSO

[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek)[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#var_second)

**dividends.future\_amount**

Returns the payment amount of the upcoming dividend in the currency of the current instrument, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_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

[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#var_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

[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings)

**earnings.future\_revenue**

Returns the estimated Revenue of the next earnings report in the currency of the instrument, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_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

[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings)

**earnings.future\_time**

Returns a UNIX timestamp indicating the expected time of the next earnings report, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_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

[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[open](https://www.tradingview.com/pine-script-reference/v6/#var_open)[low](https://www.tradingview.com/pine-script-reference/v6/#var_low)[close](https://www.tradingview.com/pine-script-reference/v6/#var_close)[volume](https://www.tradingview.com/pine-script-reference/v6/#var_volume)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[hl2](https://www.tradingview.com/pine-script-reference/v6/#var_hl2)[hlc3](https://www.tradingview.com/pine-script-reference/v6/#var_hlc3)[hlcc4](https://www.tradingview.com/pine-script-reference/v6/#var_hlcc4)[ohlc4](https://www.tradingview.com/pine-script-reference/v6/#var_ohlc4)

**hl2**

Is a shortcut for (high + low)/2

TYPE

series float

SEE ALSO

[open](https://www.tradingview.com/pine-script-reference/v6/#var_open)[high](https://www.tradingview.com/pine-script-reference/v6/#var_high)[low](https://www.tradingview.com/pine-script-reference/v6/#var_low)[close](https://www.tradingview.com/pine-script-reference/v6/#var_close)[volume](https://www.tradingview.com/pine-script-reference/v6/#var_volume)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[hlc3](https://www.tradingview.com/pine-script-reference/v6/#var_hlc3)[hlcc4](https://www.tradingview.com/pine-script-reference/v6/#var_hlcc4)[ohlc4](https://www.tradingview.com/pine-script-reference/v6/#var_ohlc4)

**hlc3**

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

TYPE

series float

SEE ALSO

[open](https://www.tradingview.com/pine-script-reference/v6/#var_open)[high](https://www.tradingview.com/pine-script-reference/v6/#var_high)[low](https://www.tradingview.com/pine-script-reference/v6/#var_low)[close](https://www.tradingview.com/pine-script-reference/v6/#var_close)[volume](https://www.tradingview.com/pine-script-reference/v6/#var_volume)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[hl2](https://www.tradingview.com/pine-script-reference/v6/#var_hl2)[hlcc4](https://www.tradingview.com/pine-script-reference/v6/#var_hlcc4)[ohlc4](https://www.tradingview.com/pine-script-reference/v6/#var_ohlc4)

**hlcc4**

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

TYPE

series float

SEE ALSO

[open](https://www.tradingview.com/pine-script-reference/v6/#var_open)[high](https://www.tradingview.com/pine-script-reference/v6/#var_high)[low](https://www.tradingview.com/pine-script-reference/v6/#var_low)[close](https://www.tradingview.com/pine-script-reference/v6/#var_close)[volume](https://www.tradingview.com/pine-script-reference/v6/#var_volume)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[hl2](https://www.tradingview.com/pine-script-reference/v6/#var_hl2)[hlc3](https://www.tradingview.com/pine-script-reference/v6/#var_hlc3)[ohlc4](https://www.tradingview.com/pine-script-reference/v6/#var_ohlc4)

**hour**

Current bar hour in exchange timezone.

TYPE

series int

SEE ALSO

[hour](https://www.tradingview.com/pine-script-reference/v6/#fun_hour)[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#var_second)

**label.all**

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

TYPE

array<label>

EXAMPLE

//**@version=**6

indicator("label.all")

//delete all labels

label.new(bar\_index, close)

a\_allLabels = label.all

if array.size(a\_allLabels) > 0

for i = 0 to array.size(a\_allLabels) - 1

label.delete(array.get(a\_allLabels, i))

REMARKS

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

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[line.all](https://www.tradingview.com/pine-script-reference/v6/#var_line.all)[box.all](https://www.tradingview.com/pine-script-reference/v6/#var_box.all)[table.all](https://www.tradingview.com/pine-script-reference/v6/#var_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

//**@version=**6

strategy("Mark Last X Bars For Backtesting", overlay = true, calc\_on\_every\_tick = true)

lastBarsFilterInput = input.int(100, "Bars Count:")

// Here, we store the 'last\_bar\_index' value that is known from the beginning of the script's calculation.

// The 'last\_bar\_index' will change when new real-time bars appear, so we declare 'lastbar' with the 'var' keyword.

var lastbar = last\_bar\_index

// Check if the current bar\_index is 'lastBarsFilterInput' removed from the last bar on the chart, or the chart is traded in real-time.

allowedToTrade = (lastbar - bar\_index <= lastBarsFilterInput) or barstate.isrealtime

bgcolor(allowedToTrade ? color.new(color.green, 80) : na)

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](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[bar\_index](https://www.tradingview.com/pine-script-reference/v6/#var_bar_index)[last\_bar\_time](https://www.tradingview.com/pine-script-reference/v6/#var_last_bar_time)[barstate.ishistory](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.ishistory)[barstate.isrealtime](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

Note that this variable returns the timestamp based on the time of the bar's open.

SEE ALSO

[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[timenow](https://www.tradingview.com/pine-script-reference/v6/#var_timenow)[timestamp](https://www.tradingview.com/pine-script-reference/v6/#fun_timestamp)[last\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#var_last_bar_index)

**line.all**

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

TYPE

array<line>

EXAMPLE

//**@version=**6

indicator("line.all")

//delete all lines

line.new(bar\_index - 10, close, bar\_index, close)

a\_allLines = line.all

if array.size(a\_allLines) > 0

for i = 0 to array.size(a\_allLines) - 1

line.delete(array.get(a\_allLines, i))

REMARKS

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

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[label.all](https://www.tradingview.com/pine-script-reference/v6/#var_label.all)[box.all](https://www.tradingview.com/pine-script-reference/v6/#var_box.all)[table.all](https://www.tradingview.com/pine-script-reference/v6/#var_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

[open](https://www.tradingview.com/pine-script-reference/v6/#var_open)[high](https://www.tradingview.com/pine-script-reference/v6/#var_high)[close](https://www.tradingview.com/pine-script-reference/v6/#var_close)[volume](https://www.tradingview.com/pine-script-reference/v6/#var_volume)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[hl2](https://www.tradingview.com/pine-script-reference/v6/#var_hl2)[hlc3](https://www.tradingview.com/pine-script-reference/v6/#var_hlc3)[hlcc4](https://www.tradingview.com/pine-script-reference/v6/#var_hlcc4)[ohlc4](https://www.tradingview.com/pine-script-reference/v6/#var_ohlc4)

**minute**

Current bar minute in exchange timezone.

TYPE

series int

SEE ALSO

[minute](https://www.tradingview.com/pine-script-reference/v6/#fun_minute)[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[second](https://www.tradingview.com/pine-script-reference/v6/#var_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

[month](https://www.tradingview.com/pine-script-reference/v6/#fun_month)[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#var_second)

**na**

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

TYPE

simple na

EXAMPLE

//**@version=**6

indicator("na")

// CORRECT

// Plot no value when on bars zero to nine. Plot `close` on other bars.

plot(bar\_index < 10 ? na : close)

// CORRECT ALTERNATIVE

// Initialize `a` to `na`. Reassign `close` to `a` on bars 10 and later.

**float** a = na

if bar\_index >= 10

a := close

plot(a)

// INCORRECT

// Trying to test the preceding bar's `close` for `na`.

// The next line, if uncommented, will cause a compilation error, because direct comparison with `na` is not allowed.

// plot(close[1] == na ? close : close[1])

// CORRECT

// Use the `na()` function to test for `na`.

plot(na(close[1]) ? close : close[1])

// CORRECT ALTERNATIVE

// `nz()` tests `close[1]` for `na`. It returns `close[1]` if it is not `na`, and `close` if it is.

plot(nz(close[1], close))

REMARKS

Do not use this variable with [comparison operators](https://www.tradingview.com/pine-script-docs/language/operators/#comparison-operators) to test values for na, as it might lead to unexpected behavior. Instead, use the [na](https://www.tradingview.com/pine-script-reference/v6/#fun_na) function. Note that na can be used to initialize variables when the initialization statement also specifies the variable's type.

SEE ALSO

[na](https://www.tradingview.com/pine-script-reference/v6/#fun_na)[nz](https://www.tradingview.com/pine-script-reference/v6/#fun_nz)[fixnan](https://www.tradingview.com/pine-script-reference/v6/#fun_fixnan)

**ohlc4**

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

TYPE

series float

SEE ALSO

[open](https://www.tradingview.com/pine-script-reference/v6/#var_open)[high](https://www.tradingview.com/pine-script-reference/v6/#var_high)[low](https://www.tradingview.com/pine-script-reference/v6/#var_low)[close](https://www.tradingview.com/pine-script-reference/v6/#var_close)[volume](https://www.tradingview.com/pine-script-reference/v6/#var_volume)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[hl2](https://www.tradingview.com/pine-script-reference/v6/#var_hl2)[hlc3](https://www.tradingview.com/pine-script-reference/v6/#var_hlc3)[hlcc4](https://www.tradingview.com/pine-script-reference/v6/#var_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

[high](https://www.tradingview.com/pine-script-reference/v6/#var_high)[low](https://www.tradingview.com/pine-script-reference/v6/#var_low)[close](https://www.tradingview.com/pine-script-reference/v6/#var_close)[volume](https://www.tradingview.com/pine-script-reference/v6/#var_volume)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[hl2](https://www.tradingview.com/pine-script-reference/v6/#var_hl2)[hlc3](https://www.tradingview.com/pine-script-reference/v6/#var_hlc3)[hlcc4](https://www.tradingview.com/pine-script-reference/v6/#var_hlcc4)[ohlc4](https://www.tradingview.com/pine-script-reference/v6/#var_ohlc4)

**polyline.all**

Returns an array containing all current [polyline](https://www.tradingview.com/pine-script-reference/v6/#type_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

[second](https://www.tradingview.com/pine-script-reference/v6/#fun_second)[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)

**session.isfirstbar**

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the current bar is the first bar of the day's session, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise. If extended session information is used, only returns true on the first bar of the pre-market bars.

TYPE

series bool

EXAMPLE

//**@version=**6

strategy("`session.isfirstbar` Example", overlay = true)

longCondition = year >= 2022

// Place a long order at the `close` of the trading session's first bar.

if session.isfirstbar and longCondition

strategy.entry("Long", strategy.long)

// Close the long position at the `close` of the trading session's last bar.

if session.islastbar and barstate.isconfirmed

strategy.close("Long", immediately = true)

SEE ALSO

[session.isfirstbar\_regular](https://www.tradingview.com/pine-script-reference/v6/#var_session.isfirstbar_regular)[session.islastbar](https://www.tradingview.com/pine-script-reference/v6/#var_session.islastbar)[session.islastbar\_regular](https://www.tradingview.com/pine-script-reference/v6/#var_session.islastbar_regular)

**session.isfirstbar\_regular**

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) on the first regular session bar of the day, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise. The result is the same whether extended session information is used or not.

TYPE

series bool

EXAMPLE

//**@version=**6

strategy("`session.isfirstbar\_regular` Example", overlay = true)

longCondition = year >= 2022

// Place a long order at the `close` of the trading session's first bar.

if session.isfirstbar and longCondition

strategy.entry("Long", strategy.long)

// Close the long position at the `close` of the trading session's last bar.

if session.islastbar\_regular and barstate.isconfirmed

strategy.close("Long", immediately = true)

SEE ALSO

[session.isfirstbar](https://www.tradingview.com/pine-script-reference/v6/#var_session.isfirstbar)[session.islastbar](https://www.tradingview.com/pine-script-reference/v6/#var_session.islastbar)

**session.islastbar**

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the current bar is the last bar of the day's session, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise. If extended session information is used, only returns true on the last bar of the post-market bars.

TYPE

series bool

EXAMPLE

//**@version=**6

strategy("`session.islastbar` Example", overlay = true)

longCondition = year >= 2022

// Place a long order at the `close` of the trading session's last bar.

// The position will enter on the `open` of next session's first bar.

if session.islastbar and longCondition

strategy.entry("Long", strategy.long)

// Close 'Long' position at the close of the last bar of the trading session

if session.islastbar and barstate.isconfirmed

strategy.close("Long", immediately = true)

REMARKS

This variable is not guaranteed to return [true](https://www.tradingview.com/pine-script-reference/v6/#const_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.

This variable is not guaranteed to work as expected on non-standard chart types, e.g., Renko.

SEE ALSO

[session.isfirstbar](https://www.tradingview.com/pine-script-reference/v6/#var_session.isfirstbar)[session.islastbar\_regular](https://www.tradingview.com/pine-script-reference/v6/#var_session.islastbar_regular)

**session.islastbar\_regular**

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) on the last regular session bar of the day, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise. The result is the same whether extended session information is used or not.

TYPE

series bool

EXAMPLE

//**@version=**6

strategy("`session.islastbar\_regular` Example", overlay = true)

longCondition = year >= 2022

// Place a long order at the `close` of the trading session's first bar.

if session.isfirstbar and longCondition

strategy.entry("Long", strategy.long)

// Close the long position at the `close` of the trading session's last bar.

if session.islastbar\_regular and barstate.isconfirmed

strategy.close("Long", immediately = true)

REMARKS

This variable is not guaranteed to return [true](https://www.tradingview.com/pine-script-reference/v6/#const_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.

This variable is not guaranteed to work as expected on non-standard chart types, e.g., Renko.

SEE ALSO

[session.isfirstbar](https://www.tradingview.com/pine-script-reference/v6/#var_session.isfirstbar)[session.islastbar](https://www.tradingview.com/pine-script-reference/v6/#var_session.islastbar)[session.isfirstbar\_regular](https://www.tradingview.com/pine-script-reference/v6/#var_session.isfirstbar_regular)

**session.ismarket**

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the current bar is a part of the regular trading hours (i.e. market hours), [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

TYPE

series bool

SEE ALSO

[session.ispremarket](https://www.tradingview.com/pine-script-reference/v6/#var_session.ispremarket)[session.ispostmarket](https://www.tradingview.com/pine-script-reference/v6/#var_session.ispostmarket)

**session.ispostmarket**

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the current bar is a part of the post-market, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise. On non-intraday charts always returns false.

TYPE

series bool

SEE ALSO

[session.ismarket](https://www.tradingview.com/pine-script-reference/v6/#var_session.ismarket)[session.ispremarket](https://www.tradingview.com/pine-script-reference/v6/#var_session.ispremarket)

**session.ispremarket**

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the current bar is a part of the pre-market, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise. On non-intraday charts always returns false.

TYPE

series bool

SEE ALSO

[session.ismarket](https://www.tradingview.com/pine-script-reference/v6/#var_session.ismarket)[session.ispostmarket](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)[strategy.convert\_to\_account](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.convert_to_account)[strategy.convert\_to\_symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[strategy.avg\_losing\_trade\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.avg\_losing\_trade](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.avg\_trade\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.avg\_trade](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.avg\_winning\_trade\_percent](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.avg\_winning\_trade](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.avg_winning_trade)

**strategy.closedtrades**

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

TYPE

series int

SEE ALSO

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.opentrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.opentrades)[strategy.wintrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.wintrades)[strategy.losstrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.losstrades)[strategy.eventrades](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.opentrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.opentrades)[strategy.wintrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.wintrades)[strategy.losstrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.losstrades)[strategy.eventrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.eventrades)

**strategy.equity**

Current equity ([strategy.initial\_capital](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.initial_capital) + [strategy.netprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit) + [strategy.openprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.openprofit)).

TYPE

series float

SEE ALSO

[strategy.netprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit)[strategy.openprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.openprofit)[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)

**strategy.eventrades**

Number of breakeven trades for the whole trading range.

TYPE

series int

SEE ALSO

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.opentrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.opentrades)[strategy.closedtrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.closedtrades)[strategy.wintrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.wintrades)[strategy.losstrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.losstrades)

**strategy.grossloss**

Total currency value of all completed losing trades.

TYPE

series float

SEE ALSO

[strategy.netprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit)[strategy.grossprofit](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.grossloss](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.grossloss)

**strategy.grossprofit**

Total currency value of all completed winning trades.

TYPE

series float

SEE ALSO

[strategy.netprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit)[strategy.grossloss](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.grossprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.grossprofit)

**strategy.initial\_capital**

The amount of initial capital set in the strategy properties.

TYPE

series float

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**strategy.losstrades**

Number of unprofitable trades for the whole trading range.

TYPE

series int

SEE ALSO

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.opentrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.opentrades)[strategy.closedtrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.closedtrades)[strategy.wintrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.wintrades)[strategy.eventrades](https://www.tradingview.com/pine-script-reference/v6/#var_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

//**@version=**6

strategy("Margin call management", overlay = true, margin\_long = 25, margin\_short = 25,

default\_qty\_type = strategy.percent\_of\_equity, default\_qty\_value = 395)

**float** maFast = ta.sma(close, 14)

**float** maSlow = ta.sma(close, 28)

if ta.crossover(maFast, maSlow)

strategy.entry("Long", strategy.long)

if ta.crossunder(maFast, maSlow)

strategy.entry("Short", strategy.short)

changePercent(v1, v2) =>

**float** result = (v1 - v2) \* 100 / math.abs(v2)

// exit when we're 10% away from a margin call, to prevent it.

if math.abs(changePercent(close, strategy.margin\_liquidation\_price)) <= 10

strategy.close("Long")

strategy.close("Short")

REMARKS

The variable returns [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) if the strategy does not use margin, i.e., the [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.max\_contracts\_held\_long](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_contracts_held_long)[strategy.max\_contracts\_held\_short](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.max\_contracts\_held\_all](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_contracts_held_all)[strategy.max\_contracts\_held\_short](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.max\_contracts\_held\_all](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_contracts_held_all)[strategy.max\_contracts\_held\_long](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_contracts_held_long)

**strategy.max\_drawdown**

Maximum equity drawdown value for the whole trading range.

TYPE

series float

SEE ALSO

[strategy.netprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit)[strategy.equity](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.equity)[strategy.max\_runup](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_drawdown)

**strategy.max\_runup**

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

TYPE

series float

SEE ALSO

[strategy.netprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit)[strategy.equity](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.equity)[strategy.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.max\_runup](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_runup)

**strategy.netprofit**

Total currency value of all completed trades.

TYPE

series float

SEE ALSO

[strategy.openprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.openprofit)[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.grossprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.grossprofit)[strategy.grossloss](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.netprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit)

**strategy.openprofit**

Current unrealized profit or loss for all open positions.

TYPE

series float

SEE ALSO

[strategy.netprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit)[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.openprofit](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)

**strategy.opentrades.capital\_held**

Returns the capital amount currently held by open trades.

TYPE

series float

EXAMPLE

//**@version=**6

strategy(

"strategy.opentrades.capital\_held example", overlay=false, margin\_long=50, margin\_short=50,

default\_qty\_type = strategy.percent\_of\_equity, default\_qty\_value = 100

)

// Enter a short position on the first bar.

if barstate.isfirst

strategy.entry("Short", strategy.short)

// Plot the capital held by the short position.

plot(strategy.opentrades.capital\_held, "Capital held")

// Highlight the chart background if the position is completely closed by margin calls.

bgcolor(bar\_index > 0 and strategy.opentrades.capital\_held == 0 ? color.new(color.red, 60) : na)

REMARKS

This variable returns [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) if the strategy does not simulate funding trades with a portion of the hypothetical account, i.e., if the [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)

**strategy.position\_entry\_name**

Name of the order that initially opened current market position.

TYPE

series string

SEE ALSO

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_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

[strategy.position\_avg\_price](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_avg_price)

**strategy.wintrades**

Number of profitable trades for the whole trading range.

TYPE

series int

SEE ALSO

[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.opentrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.opentrades)[strategy.closedtrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.closedtrades)[strategy.losstrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.losstrades)[strategy.eventrades](https://www.tradingview.com/pine-script-reference/v6/#var_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

[syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)

**syminfo.country**

Returns the two-letter code of the country where the symbol is traded, in the [ISO 3166-1 alpha-2](https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2) format, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_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

[syminfo.basecurrency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.basecurrency)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[currency.USD](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USD)[currency.EUR](https://www.tradingview.com/pine-script-reference/v6/#const_currency.EUR)

**syminfo.description**

Description for the current symbol.

TYPE

simple string

SEE ALSO

[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.prefix](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.prefix)

**syminfo.employees**

The number of employees the company has.

TYPE

simple int

EXAMPLE

//**@version=**6

indicator("syminfo simple")

//**@variable** A table containing information about a company's employees, shareholders, and shares.

var result\_table = table.new(position = position.top\_right, columns = 2, rows = 5, border\_width = 1)

if barstate.islastconfirmedhistory

// Add header cells

table.cell(table\_id = result\_table, column = 0, row = 0, text = "name")

table.cell(table\_id = result\_table, column = 1, row = 0, text = "value")

// Add employee info cells.

table.cell(table\_id = result\_table, column = 0, row = 1, text = "employees")

table.cell(table\_id = result\_table, column = 1, row = 1, text = str.tostring(syminfo.employees))

// Add shareholder cells.

table.cell(table\_id = result\_table, column = 0, row = 2, text = "shareholders")

table.cell(table\_id = result\_table, column = 1, row = 2, text = str.tostring(syminfo.shareholders))

// Add float shares outstanding cells.

table.cell(table\_id = result\_table, column = 0, row = 3, text = "shares\_outstanding\_float")

table.cell(table\_id = result\_table, column = 1, row = 3, text = str.tostring(syminfo.shares\_outstanding\_float))

// Add total shares outstanding cells.

table.cell(table\_id = result\_table, column = 0, row = 4, text = "shares\_outstanding\_total")

table.cell(table\_id = result\_table, column = 1, row = 4, text = str.tostring(syminfo.shares\_outstanding\_total))

SEE ALSO

[syminfo.shareholders](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.shareholders)[syminfo.shares\_outstanding\_float](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.shares_outstanding_float)[syminfo.shares\_outstanding\_total](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_na).

TYPE

simple int

**syminfo.industry**

Returns the industry of the symbol, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_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

[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[timeframe.main\_period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.main_period)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period)[timeframe.multiplier](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.multiplier)[syminfo.root](https://www.tradingview.com/pine-script-reference/v6/#var_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

[syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mintick)[syminfo.pointvalue](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.pointvalue)

**syminfo.minmove**

Returns a whole number used to calculate the smallest increment between a symbol's price movements ([syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mintick)). It is the numerator in the [syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mintick) formula: syminfo.minmove / syminfo.pricescale = syminfo.mintick.

TYPE

simple int

SEE ALSO

[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period)[timeframe.multiplier](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.multiplier)[syminfo.root](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.root)

**syminfo.mintick**

Min tick value for the current symbol.

TYPE

simple float

SEE ALSO

[syminfo.pointvalue](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.pointvalue)[syminfo.mincontract](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mincontract)

**syminfo.pointvalue**

Point value for the current symbol.

TYPE

simple float

SEE ALSO

[syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mintick)[syminfo.mincontract](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mincontract)

**syminfo.prefix**

Prefix of current symbol name (i.e. for 'CME\_EOD:TICKER' prefix is 'CME\_EOD').

TYPE

simple string

EXAMPLE

//**@version=**6

indicator("syminfo.prefix")

// If current chart symbol is 'BATS:MSFT' then syminfo.prefix is 'BATS'.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, text=syminfo.prefix)

SEE ALSO

[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)

**syminfo.pricescale**

Returns a whole number used to calculate the smallest increment between a symbol's price movements ([syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mintick)). It is the denominator in the [syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mintick) formula: syminfo.minmove / syminfo.pricescale = syminfo.mintick.

TYPE

simple int

SEE ALSO

[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period)[timeframe.multiplier](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.multiplier)[syminfo.root](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.root)

**syminfo.recommendations\_buy**

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

TYPE

series int

EXAMPLE

//**@version=**6

indicator("syminfo recommendations", overlay = true)

//**@variable** A table containing information about analyst recommendations.

var **table** ratings = table.new(position.top\_right, 8, 2, frame\_color = #000000)

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

// Add header cells.

table.cell(ratings, 0, 0, "Start Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 0, "End Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 0, "Buy", bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 0, "Strong Buy", bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 0, "Sell", bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 5, 0, "Strong Sell", bgcolor = color.red, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 6, 0, "Hold", bgcolor = color.orange, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 7, 0, "Total", bgcolor = color.silver, text\_color = #000000, text\_size = size.large)

// Recommendation strings

**string** startDate = str.format\_time(syminfo.recommendations\_date, "yyyy-MM-dd")

**string** endDate = str.format\_time(YTD, "yyyy-MM-dd")

**string** buyRatings = str.tostring(syminfo.recommendations\_buy)

**string** strongBuyRatings = str.tostring(syminfo.recommendations\_buy\_strong)

**string** sellRatings = str.tostring(syminfo.recommendations\_sell)

**string** strongSellRatings = str.tostring(syminfo.recommendations\_sell\_strong)

**string** holdRatings = str.tostring(syminfo.recommendations\_hold)

**string** totalRatings = str.tostring(syminfo.recommendations\_total)

// Add value cells

table.cell(ratings, 0, 1, startDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 1, endDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 1, buyRatings, bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 1, strongBuyRatings, bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 1, sellRatings, bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

SEE ALSO

[syminfo.recommendations\_buy\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy_strong)[syminfo.recommendations\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_date)[syminfo.recommendations\_hold](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_hold)[syminfo.recommendations\_total](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_total)[syminfo.recommendations\_sell](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell)[syminfo.recommendations\_sell\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_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

//**@version=**6

indicator("syminfo recommendations", overlay = true)

//**@variable** A table containing information about analyst recommendations.

var **table** ratings = table.new(position.top\_right, 8, 2, frame\_color = #000000)

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

// Add header cells.

table.cell(ratings, 0, 0, "Start Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 0, "End Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 0, "Buy", bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 0, "Strong Buy", bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 0, "Sell", bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 5, 0, "Strong Sell", bgcolor = color.red, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 6, 0, "Hold", bgcolor = color.orange, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 7, 0, "Total", bgcolor = color.silver, text\_color = #000000, text\_size = size.large)

// Recommendation strings

**string** startDate = str.format\_time(syminfo.recommendations\_date, "yyyy-MM-dd")

**string** endDate = str.format\_time(YTD, "yyyy-MM-dd")

**string** buyRatings = str.tostring(syminfo.recommendations\_buy)

**string** strongBuyRatings = str.tostring(syminfo.recommendations\_buy\_strong)

**string** sellRatings = str.tostring(syminfo.recommendations\_sell)

**string** strongSellRatings = str.tostring(syminfo.recommendations\_sell\_strong)

**string** holdRatings = str.tostring(syminfo.recommendations\_hold)

**string** totalRatings = str.tostring(syminfo.recommendations\_total)

// Add value cells

table.cell(ratings, 0, 1, startDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 1, endDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 1, buyRatings, bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 1, strongBuyRatings, bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 1, sellRatings, bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

SEE ALSO

[syminfo.recommendations\_buy](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy)[syminfo.recommendations\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_date)[syminfo.recommendations\_hold](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_hold)[syminfo.recommendations\_total](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_total)[syminfo.recommendations\_sell](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell)[syminfo.recommendations\_sell\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell_strong)

**syminfo.recommendations\_date**

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

TYPE

series int

EXAMPLE

//**@version=**6

indicator("syminfo recommendations", overlay = true)

//**@variable** A table containing information about analyst recommendations.

var **table** ratings = table.new(position.top\_right, 8, 2, frame\_color = #000000)

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

// Add header cells.

table.cell(ratings, 0, 0, "Start Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 0, "End Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 0, "Buy", bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 0, "Strong Buy", bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 0, "Sell", bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 5, 0, "Strong Sell", bgcolor = color.red, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 6, 0, "Hold", bgcolor = color.orange, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 7, 0, "Total", bgcolor = color.silver, text\_color = #000000, text\_size = size.large)

// Recommendation strings

**string** startDate = str.format\_time(syminfo.recommendations\_date, "yyyy-MM-dd")

**string** endDate = str.format\_time(YTD, "yyyy-MM-dd")

**string** buyRatings = str.tostring(syminfo.recommendations\_buy)

**string** strongBuyRatings = str.tostring(syminfo.recommendations\_buy\_strong)

**string** sellRatings = str.tostring(syminfo.recommendations\_sell)

**string** strongSellRatings = str.tostring(syminfo.recommendations\_sell\_strong)

**string** holdRatings = str.tostring(syminfo.recommendations\_hold)

**string** totalRatings = str.tostring(syminfo.recommendations\_total)

// Add value cells

table.cell(ratings, 0, 1, startDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 1, endDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 1, buyRatings, bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 1, strongBuyRatings, bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 1, sellRatings, bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

SEE ALSO

[syminfo.recommendations\_buy](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy)[syminfo.recommendations\_buy\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy_strong)[syminfo.recommendations\_hold](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_hold)[syminfo.recommendations\_total](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_total)[syminfo.recommendations\_sell](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell)[syminfo.recommendations\_sell\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell_strong)

**syminfo.recommendations\_hold**

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

TYPE

series int

EXAMPLE

//**@version=**6

indicator("syminfo recommendations", overlay = true)

//**@variable** A table containing information about analyst recommendations.

var **table** ratings = table.new(position.top\_right, 8, 2, frame\_color = #000000)

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

// Add header cells.

table.cell(ratings, 0, 0, "Start Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 0, "End Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 0, "Buy", bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 0, "Strong Buy", bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 0, "Sell", bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 5, 0, "Strong Sell", bgcolor = color.red, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 6, 0, "Hold", bgcolor = color.orange, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 7, 0, "Total", bgcolor = color.silver, text\_color = #000000, text\_size = size.large)

// Recommendation strings

**string** startDate = str.format\_time(syminfo.recommendations\_date, "yyyy-MM-dd")

**string** endDate = str.format\_time(YTD, "yyyy-MM-dd")

**string** buyRatings = str.tostring(syminfo.recommendations\_buy)

**string** strongBuyRatings = str.tostring(syminfo.recommendations\_buy\_strong)

**string** sellRatings = str.tostring(syminfo.recommendations\_sell)

**string** strongSellRatings = str.tostring(syminfo.recommendations\_sell\_strong)

**string** holdRatings = str.tostring(syminfo.recommendations\_hold)

**string** totalRatings = str.tostring(syminfo.recommendations\_total)

// Add value cells

table.cell(ratings, 0, 1, startDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 1, endDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 1, buyRatings, bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 1, strongBuyRatings, bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 1, sellRatings, bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

SEE ALSO

[syminfo.recommendations\_buy](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy)[syminfo.recommendations\_buy\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy_strong)[syminfo.recommendations\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_date)[syminfo.recommendations\_total](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_total)[syminfo.recommendations\_sell](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell)[syminfo.recommendations\_sell\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell_strong)

**syminfo.recommendations\_sell**

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

TYPE

series int

EXAMPLE

//**@version=**6

indicator("syminfo recommendations", overlay = true)

//**@variable** A table containing information about analyst recommendations.

var **table** ratings = table.new(position.top\_right, 8, 2, frame\_color = #000000)

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

// Add header cells.

table.cell(ratings, 0, 0, "Start Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 0, "End Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 0, "Buy", bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 0, "Strong Buy", bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 0, "Sell", bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 5, 0, "Strong Sell", bgcolor = color.red, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 6, 0, "Hold", bgcolor = color.orange, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 7, 0, "Total", bgcolor = color.silver, text\_color = #000000, text\_size = size.large)

// Recommendation strings

**string** startDate = str.format\_time(syminfo.recommendations\_date, "yyyy-MM-dd")

**string** endDate = str.format\_time(YTD, "yyyy-MM-dd")

**string** buyRatings = str.tostring(syminfo.recommendations\_buy)

**string** strongBuyRatings = str.tostring(syminfo.recommendations\_buy\_strong)

**string** sellRatings = str.tostring(syminfo.recommendations\_sell)

**string** strongSellRatings = str.tostring(syminfo.recommendations\_sell\_strong)

**string** holdRatings = str.tostring(syminfo.recommendations\_hold)

**string** totalRatings = str.tostring(syminfo.recommendations\_total)

// Add value cells

table.cell(ratings, 0, 1, startDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 1, endDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 1, buyRatings, bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 1, strongBuyRatings, bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 1, sellRatings, bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

SEE ALSO

[syminfo.recommendations\_buy](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy)[syminfo.recommendations\_buy\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy_strong)[syminfo.recommendations\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_date)[syminfo.recommendations\_hold](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_hold)[syminfo.recommendations\_total](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_total)[syminfo.recommendations\_sell\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_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

//**@version=**6

indicator("syminfo recommendations", overlay = true)

//**@variable** A table containing information about analyst recommendations.

var **table** ratings = table.new(position.top\_right, 8, 2, frame\_color = #000000)

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

// Add header cells.

table.cell(ratings, 0, 0, "Start Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 0, "End Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 0, "Buy", bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 0, "Strong Buy", bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 0, "Sell", bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 5, 0, "Strong Sell", bgcolor = color.red, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 6, 0, "Hold", bgcolor = color.orange, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 7, 0, "Total", bgcolor = color.silver, text\_color = #000000, text\_size = size.large)

// Recommendation strings

**string** startDate = str.format\_time(syminfo.recommendations\_date, "yyyy-MM-dd")

**string** endDate = str.format\_time(YTD, "yyyy-MM-dd")

**string** buyRatings = str.tostring(syminfo.recommendations\_buy)

**string** strongBuyRatings = str.tostring(syminfo.recommendations\_buy\_strong)

**string** sellRatings = str.tostring(syminfo.recommendations\_sell)

**string** strongSellRatings = str.tostring(syminfo.recommendations\_sell\_strong)

**string** holdRatings = str.tostring(syminfo.recommendations\_hold)

**string** totalRatings = str.tostring(syminfo.recommendations\_total)

// Add value cells

table.cell(ratings, 0, 1, startDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 1, endDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 1, buyRatings, bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 1, strongBuyRatings, bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 1, sellRatings, bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

SEE ALSO

[syminfo.recommendations\_buy](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy)[syminfo.recommendations\_buy\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy_strong)[syminfo.recommendations\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_date)[syminfo.recommendations\_hold](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_hold)[syminfo.recommendations\_total](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_total)[syminfo.recommendations\_sell](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell)

**syminfo.recommendations\_total**

The total number of recommendations for the current symbol.

TYPE

series int

EXAMPLE

//**@version=**6

indicator("syminfo recommendations", overlay = true)

//**@variable** A table containing information about analyst recommendations.

var **table** ratings = table.new(position.top\_right, 8, 2, frame\_color = #000000)

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

// Add header cells.

table.cell(ratings, 0, 0, "Start Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 0, "End Date", bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 0, "Buy", bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 0, "Strong Buy", bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 0, "Sell", bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 5, 0, "Strong Sell", bgcolor = color.red, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 6, 0, "Hold", bgcolor = color.orange, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 7, 0, "Total", bgcolor = color.silver, text\_color = #000000, text\_size = size.large)

// Recommendation strings

**string** startDate = str.format\_time(syminfo.recommendations\_date, "yyyy-MM-dd")

**string** endDate = str.format\_time(YTD, "yyyy-MM-dd")

**string** buyRatings = str.tostring(syminfo.recommendations\_buy)

**string** strongBuyRatings = str.tostring(syminfo.recommendations\_buy\_strong)

**string** sellRatings = str.tostring(syminfo.recommendations\_sell)

**string** strongSellRatings = str.tostring(syminfo.recommendations\_sell\_strong)

**string** holdRatings = str.tostring(syminfo.recommendations\_hold)

**string** totalRatings = str.tostring(syminfo.recommendations\_total)

// Add value cells

table.cell(ratings, 0, 1, startDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 1, 1, endDate, bgcolor = color.gray, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 2, 1, buyRatings, bgcolor = color.teal, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 3, 1, strongBuyRatings, bgcolor = color.lime, text\_color = #000000, text\_size = size.large)

table.cell(ratings, 4, 1, sellRatings, bgcolor = color.maroon, text\_color = #000000, text\_size = size.large)

SEE ALSO

[syminfo.recommendations\_buy](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy)[syminfo.recommendations\_buy\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_buy_strong)[syminfo.recommendations\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_date)[syminfo.recommendations\_hold](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_hold)[syminfo.recommendations\_sell](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell)[syminfo.recommendations\_sell\_strong](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.recommendations_sell_strong)

**syminfo.root**

Root for derivatives like futures contract. For other symbols returns the same value as [syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker).

TYPE

simple string

EXAMPLE

//**@version=**6

indicator("syminfo.root")

// If the current chart symbol is continuous futures ('ES1!'), it would display 'ES'.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, syminfo.root)

SEE ALSO

[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)

**syminfo.sector**

Returns the sector of the symbol, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#const_session.regular), [session.extended](https://www.tradingview.com/pine-script-reference/v6/#const_session.extended).

TYPE

simple string

SEE ALSO

[session.regular](https://www.tradingview.com/pine-script-reference/v6/#const_session.regular)[session.extended](https://www.tradingview.com/pine-script-reference/v6/#const_session.extended)

**syminfo.shareholders**

The number of shareholders the company has.

TYPE

simple int

EXAMPLE

//**@version=**6

indicator("syminfo simple")

//**@variable** A table containing information about a company's employees, shareholders, and shares.

var result\_table = table.new(position = position.top\_right, columns = 2, rows = 5, border\_width = 1)

if barstate.islastconfirmedhistory

// Add header cells

table.cell(table\_id = result\_table, column = 0, row = 0, text = "name")

table.cell(table\_id = result\_table, column = 1, row = 0, text = "value")

// Add employee info cells.

table.cell(table\_id = result\_table, column = 0, row = 1, text = "employees")

table.cell(table\_id = result\_table, column = 1, row = 1, text = str.tostring(syminfo.employees))

// Add shareholder cells.

table.cell(table\_id = result\_table, column = 0, row = 2, text = "shareholders")

table.cell(table\_id = result\_table, column = 1, row = 2, text = str.tostring(syminfo.shareholders))

// Add float shares outstanding cells.

table.cell(table\_id = result\_table, column = 0, row = 3, text = "shares\_outstanding\_float")

table.cell(table\_id = result\_table, column = 1, row = 3, text = str.tostring(syminfo.shares\_outstanding\_float))

// Add total shares outstanding cells.

table.cell(table\_id = result\_table, column = 0, row = 4, text = "shares\_outstanding\_total")

table.cell(table\_id = result\_table, column = 1, row = 4, text = str.tostring(syminfo.shares\_outstanding\_total))

SEE ALSO

[syminfo.employees](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.employees)[syminfo.shares\_outstanding\_float](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.shares_outstanding_float)[syminfo.shares\_outstanding\_total](https://www.tradingview.com/pine-script-reference/v6/#var_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

//**@version=**6

indicator("syminfo simple")

//**@variable** A table containing information about a company's employees, shareholders, and shares.

var result\_table = table.new(position = position.top\_right, columns = 2, rows = 5, border\_width = 1)

if barstate.islastconfirmedhistory

// Add header cells

table.cell(table\_id = result\_table, column = 0, row = 0, text = "name")

table.cell(table\_id = result\_table, column = 1, row = 0, text = "value")

// Add employee info cells.

table.cell(table\_id = result\_table, column = 0, row = 1, text = "employees")

table.cell(table\_id = result\_table, column = 1, row = 1, text = str.tostring(syminfo.employees))

// Add shareholder cells.

table.cell(table\_id = result\_table, column = 0, row = 2, text = "shareholders")

table.cell(table\_id = result\_table, column = 1, row = 2, text = str.tostring(syminfo.shareholders))

// Add float shares outstanding cells.

table.cell(table\_id = result\_table, column = 0, row = 3, text = "shares\_outstanding\_float")

table.cell(table\_id = result\_table, column = 1, row = 3, text = str.tostring(syminfo.shares\_outstanding\_float))

// Add total shares outstanding cells.

table.cell(table\_id = result\_table, column = 0, row = 4, text = "shares\_outstanding\_total")

table.cell(table\_id = result\_table, column = 1, row = 4, text = str.tostring(syminfo.shares\_outstanding\_total))

SEE ALSO

[syminfo.employees](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.employees)[syminfo.shareholders](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.shareholders)[syminfo.shares\_outstanding\_total](https://www.tradingview.com/pine-script-reference/v6/#var_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

//**@version=**6

indicator("syminfo simple")

//**@variable** A table containing information about a company's employees, shareholders, and shares.

var result\_table = table.new(position = position.top\_right, columns = 2, rows = 5, border\_width = 1)

if barstate.islastconfirmedhistory

// Add header cells

table.cell(table\_id = result\_table, column = 0, row = 0, text = "name")

table.cell(table\_id = result\_table, column = 1, row = 0, text = "value")

// Add employee info cells.

table.cell(table\_id = result\_table, column = 0, row = 1, text = "employees")

table.cell(table\_id = result\_table, column = 1, row = 1, text = str.tostring(syminfo.employees))

// Add shareholder cells.

table.cell(table\_id = result\_table, column = 0, row = 2, text = "shareholders")

table.cell(table\_id = result\_table, column = 1, row = 2, text = str.tostring(syminfo.shareholders))

// Add float shares outstanding cells.

table.cell(table\_id = result\_table, column = 0, row = 3, text = "shares\_outstanding\_float")

table.cell(table\_id = result\_table, column = 1, row = 3, text = str.tostring(syminfo.shares\_outstanding\_float))

// Add total shares outstanding cells.

table.cell(table\_id = result\_table, column = 0, row = 4, text = "shares\_outstanding\_total")

table.cell(table\_id = result\_table, column = 1, row = 4, text = str.tostring(syminfo.shares\_outstanding\_total))

SEE ALSO

[syminfo.employees](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.employees)[syminfo.shareholders](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.shareholders)[syminfo.shares\_outstanding\_float](https://www.tradingview.com/pine-script-reference/v6/#var_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

//**@version=**6

indicator("syminfo target\_price")

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

//**@variable** A line connecting the current `close` to the highest yearly price estimate.

highLine = line.new(time, close, YTD, syminfo.target\_price\_high, color = color.green, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the lowest yearly price estimate.

lowLine = line.new(time, close, YTD, syminfo.target\_price\_low, color = color.red, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the median yearly price estimate.

medianLine = line.new(time, close, YTD, syminfo.target\_price\_median, color = color.gray, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the average yearly price estimate.

averageLine = line.new(time, close, YTD, syminfo.target\_price\_average, color = color.orange, xloc = xloc.bar\_time)

// Fill the space between targets

linefill.new(lowLine, medianLine, color.new(color.red, 90))

linefill.new(medianLine, highLine, color.new(color.green, 90))

// Create a label displaying the total number of analyst estimates.

**string** estimatesText = str.format("Number of estimates: {0}", syminfo.target\_price\_estimates)

label.new(bar\_index, close, estimatesText, textcolor = color.white, size = size.large)

REMARKS

If analysts supply the targets when the market is closed, the variable can return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) until the market opens.

SEE ALSO

[syminfo.target\_price\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_date)[syminfo.target\_price\_estimates](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_estimates)[syminfo.target\_price\_high](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_high)[syminfo.target\_price\_low](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_low)[syminfo.target\_price\_median](https://www.tradingview.com/pine-script-reference/v6/#var_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

//**@version=**6

indicator("syminfo target\_price")

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

//**@variable** A line connecting the current `close` to the highest yearly price estimate.

highLine = line.new(time, close, YTD, syminfo.target\_price\_high, color = color.green, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the lowest yearly price estimate.

lowLine = line.new(time, close, YTD, syminfo.target\_price\_low, color = color.red, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the median yearly price estimate.

medianLine = line.new(time, close, YTD, syminfo.target\_price\_median, color = color.gray, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the average yearly price estimate.

averageLine = line.new(time, close, YTD, syminfo.target\_price\_average, color = color.orange, xloc = xloc.bar\_time)

// Fill the space between targets

linefill.new(lowLine, medianLine, color.new(color.red, 90))

linefill.new(medianLine, highLine, color.new(color.green, 90))

// Create a label displaying the total number of analyst estimates.

**string** estimatesText = str.format("Number of estimates: {0}", syminfo.target\_price\_estimates)

label.new(bar\_index, close, estimatesText, textcolor = color.white, size = size.large)

REMARKS

If analysts supply the targets when the market is closed, the variable can return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) until the market opens.

SEE ALSO

[syminfo.target\_price\_average](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_average)[syminfo.target\_price\_estimates](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_estimates)[syminfo.target\_price\_high](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_high)[syminfo.target\_price\_low](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_low)[syminfo.target\_price\_median](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_median)

**syminfo.target\_price\_estimates**

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

TYPE

series float

EXAMPLE

//**@version=**6

indicator("syminfo target\_price")

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

//**@variable** A line connecting the current `close` to the highest yearly price estimate.

highLine = line.new(time, close, YTD, syminfo.target\_price\_high, color = color.green, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the lowest yearly price estimate.

lowLine = line.new(time, close, YTD, syminfo.target\_price\_low, color = color.red, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the median yearly price estimate.

medianLine = line.new(time, close, YTD, syminfo.target\_price\_median, color = color.gray, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the average yearly price estimate.

averageLine = line.new(time, close, YTD, syminfo.target\_price\_average, color = color.orange, xloc = xloc.bar\_time)

// Fill the space between targets

linefill.new(lowLine, medianLine, color.new(color.red, 90))

linefill.new(medianLine, highLine, color.new(color.green, 90))

// Create a label displaying the total number of analyst estimates.

**string** estimatesText = str.format("Number of estimates: {0}", syminfo.target\_price\_estimates)

label.new(bar\_index, close, estimatesText, textcolor = color.white, size = size.large)

REMARKS

If analysts supply the targets when the market is closed, the variable can return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) until the market opens.

SEE ALSO

[syminfo.target\_price\_average](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_average)[syminfo.target\_price\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_date)[syminfo.target\_price\_high](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_high)[syminfo.target\_price\_low](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_low)[syminfo.target\_price\_median](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_median)

**syminfo.target\_price\_high**

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

TYPE

series float

EXAMPLE

//**@version=**6

indicator("syminfo target\_price")

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

//**@variable** A line connecting the current `close` to the highest yearly price estimate.

highLine = line.new(time, close, YTD, syminfo.target\_price\_high, color = color.green, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the lowest yearly price estimate.

lowLine = line.new(time, close, YTD, syminfo.target\_price\_low, color = color.red, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the median yearly price estimate.

medianLine = line.new(time, close, YTD, syminfo.target\_price\_median, color = color.gray, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the average yearly price estimate.

averageLine = line.new(time, close, YTD, syminfo.target\_price\_average, color = color.orange, xloc = xloc.bar\_time)

// Fill the space between targets

linefill.new(lowLine, medianLine, color.new(color.red, 90))

linefill.new(medianLine, highLine, color.new(color.green, 90))

// Create a label displaying the total number of analyst estimates.

**string** estimatesText = str.format("Number of estimates: {0}", syminfo.target\_price\_estimates)

label.new(bar\_index, close, estimatesText, textcolor = color.white, size = size.large)

REMARKS

If analysts supply the targets when the market is closed, the variable can return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) until the market opens.

SEE ALSO

[syminfo.target\_price\_average](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_average)[syminfo.target\_price\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_date)[syminfo.target\_price\_estimates](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_estimates)[syminfo.target\_price\_low](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_low)[syminfo.target\_price\_median](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_median)

**syminfo.target\_price\_low**

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

TYPE

series float

EXAMPLE

//**@version=**6

indicator("syminfo target\_price")

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

//**@variable** A line connecting the current `close` to the highest yearly price estimate.

highLine = line.new(time, close, YTD, syminfo.target\_price\_high, color = color.green, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the lowest yearly price estimate.

lowLine = line.new(time, close, YTD, syminfo.target\_price\_low, color = color.red, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the median yearly price estimate.

medianLine = line.new(time, close, YTD, syminfo.target\_price\_median, color = color.gray, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the average yearly price estimate.

averageLine = line.new(time, close, YTD, syminfo.target\_price\_average, color = color.orange, xloc = xloc.bar\_time)

// Fill the space between targets

linefill.new(lowLine, medianLine, color.new(color.red, 90))

linefill.new(medianLine, highLine, color.new(color.green, 90))

// Create a label displaying the total number of analyst estimates.

**string** estimatesText = str.format("Number of estimates: {0}", syminfo.target\_price\_estimates)

label.new(bar\_index, close, estimatesText, textcolor = color.white, size = size.large)

REMARKS

If analysts supply the targets when the market is closed, the variable can return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) until the market opens.

SEE ALSO

[syminfo.target\_price\_average](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_average)[syminfo.target\_price\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_date)[syminfo.target\_price\_estimates](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_estimates)[syminfo.target\_price\_high](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_high)[syminfo.target\_price\_median](https://www.tradingview.com/pine-script-reference/v6/#var_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

//**@version=**6

indicator("syminfo target\_price")

if barstate.islastconfirmedhistory

//**@variable** The time value one year from the date of the last analyst recommendations.

**int** YTD = syminfo.target\_price\_date + timeframe.in\_seconds("12M") \* 1000

//**@variable** A line connecting the current `close` to the highest yearly price estimate.

highLine = line.new(time, close, YTD, syminfo.target\_price\_high, color = color.green, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the lowest yearly price estimate.

lowLine = line.new(time, close, YTD, syminfo.target\_price\_low, color = color.red, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the median yearly price estimate.

medianLine = line.new(time, close, YTD, syminfo.target\_price\_median, color = color.gray, xloc = xloc.bar\_time)

//**@variable** A line connecting the current `close` to the average yearly price estimate.

averageLine = line.new(time, close, YTD, syminfo.target\_price\_average, color = color.orange, xloc = xloc.bar\_time)

// Fill the space between targets

linefill.new(lowLine, medianLine, color.new(color.red, 90))

linefill.new(medianLine, highLine, color.new(color.green, 90))

// Create a label displaying the total number of analyst estimates.

**string** estimatesText = str.format("Number of estimates: {0}", syminfo.target\_price\_estimates)

label.new(bar\_index, close, estimatesText, textcolor = color.white, size = size.large)

REMARKS

If analysts supply the targets when the market is closed, the variable can return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) until the market opens.

SEE ALSO

[syminfo.target\_price\_average](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_average)[syminfo.target\_price\_date](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_date)[syminfo.target\_price\_estimates](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_estimates)[syminfo.target\_price\_high](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_high)[syminfo.target\_price\_low](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.target_price_low)

**syminfo.ticker**

Symbol name without exchange prefix, e.g. 'MSFT'.

TYPE

simple string

SEE ALSO

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period)[timeframe.multiplier](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.multiplier)[syminfo.root](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.standard) to purify it. This will remove any extraneous information and return a ticker ID consistently formatted using the "prefix:ticker" structure.

To always access the script's main ticker ID, even within another context, use the [syminfo.main\_tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.main_tickerid) variable.

SEE ALSO

[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[syminfo.main\_tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.main_tickerid)[timeframe.main\_period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.main_period)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period)[timeframe.multiplier](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.multiplier)[syminfo.root](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.root)

**syminfo.timezone**

Timezone of the exchange of the chart main series. Possible values see in [timestamp](https://www.tradingview.com/pine-script-reference/v6/#fun_timestamp).

TYPE

simple string

SEE ALSO

[timestamp](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_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

[syminfo.type](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.type)

**ta.accdist**

Accumulation/distribution index.

TYPE

series float

**ta.iii**

Intraday Intensity Index.

TYPE

series float

EXAMPLE

//**@version=**6

indicator("Intraday Intensity Index")

plot(ta.iii, color=color.yellow)

// the same on pine

f\_iii() =>

(2 \* close - high - low) / ((high - low) \* volume)

plot(f\_iii())

**ta.nvi**

Negative Volume Index.

TYPE

series float

EXAMPLE

//**@version=**6

indicator("Negative Volume Index")

plot(ta.nvi, color=color.yellow)

// the same on pine

f\_nvi() =>

**float** ta\_nvi = 1.0

**float** prevNvi = (nz(ta\_nvi[1], 0.0) == 0.0) ? 1.0 : ta\_nvi[1]

if nz(close, 0.0) == 0.0 or nz(close[1], 0.0) == 0.0

ta\_nvi := prevNvi

else

ta\_nvi := (volume < nz(volume[1], 0.0)) ? prevNvi + ((close - close[1]) / close[1]) \* prevNvi : prevNvi

result = ta\_nvi

plot(f\_nvi())

**ta.obv**

On Balance Volume.

TYPE

series float

EXAMPLE

//**@version=**6

indicator("On Balance Volume")

plot(ta.obv, color=color.yellow)

// the same on pine

f\_obv() =>

ta.cum(math.sign(ta.change(close)) \* volume)

plot(f\_obv())

**ta.pvi**

Positive Volume Index.

TYPE

series float

EXAMPLE

//**@version=**6

indicator("Positive Volume Index")

plot(ta.pvi, color=color.yellow)

// the same on pine

f\_pvi() =>

**float** ta\_pvi = 1.0

**float** prevPvi = (nz(ta\_pvi[1], 0.0) == 0.0) ? 1.0 : ta\_pvi[1]

if nz(close, 0.0) == 0.0 or nz(close[1], 0.0) == 0.0

ta\_pvi := prevPvi

else

ta\_pvi := (volume > nz(volume[1], 0.0)) ? prevPvi + ((close - close[1]) / close[1]) \* prevPvi : prevPvi

result = ta\_pvi

plot(f\_pvi())

**ta.pvt**

Price-Volume Trend.

TYPE

series float

EXAMPLE

//**@version=**6

indicator("Price-Volume Trend")

plot(ta.pvt, color=color.yellow)

// the same on pine

f\_pvt() =>

ta.cum((ta.change(close) / close[1]) \* volume)

plot(f\_pvt())

**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

[ta.tr](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.tr)[ta.atr](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.atr)

**ta.vwap**

Volume Weighted Average Price. It uses [hlc3](https://www.tradingview.com/pine-script-reference/v6/#var_hlc3) as its source series.

TYPE

series float

SEE ALSO

[ta.vwap](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwap)

**ta.wad**

Williams Accumulation/Distribution.

TYPE

series float

EXAMPLE

//**@version=**6

indicator("Williams Accumulation/Distribution")

plot(ta.wad, color=color.yellow)

// the same on pine

f\_wad() =>

trueHigh = math.max(high, close[1])

trueLow = math.min(low, close[1])

mom = ta.change(close)

gain = (mom > 0) ? close - trueLow : (mom < 0) ? close - trueHigh : 0

ta.cum(gain)

plot(f\_wad())

**ta.wvad**

Williams Variable Accumulation/Distribution.

TYPE

series float

EXAMPLE

//**@version=**6

indicator("Williams Variable Accumulation/Distribution")

plot(ta.wvad, color=color.yellow)

// the same on pine

f\_wvad() =>

(close - open) / (high - low) \* volume

plot(f\_wvad())

**table.all**

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

TYPE

array<table>

EXAMPLE

//**@version=**6

indicator("table.all")

//delete all tables

table.new(position = position.top\_right, columns = 2, rows = 1, bgcolor = color.yellow, border\_width = 1)

a\_allTables = table.all

if array.size(a\_allTables) > 0

for i = 0 to array.size(a\_allTables) - 1

table.delete(array.get(a\_allTables, i))

REMARKS

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

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[line.all](https://www.tradingview.com/pine-script-reference/v6/#var_line.all)[label.all](https://www.tradingview.com/pine-script-reference/v6/#var_label.all)[box.all](https://www.tradingview.com/pine-script-reference/v6/#var_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

[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[time\_close](https://www.tradingview.com/pine-script-reference/v6/#var_time_close)[timenow](https://www.tradingview.com/pine-script-reference/v6/#var_timenow)[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_na) on the chart's realtime bars.

TYPE

series int

SEE ALSO

[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[timenow](https://www.tradingview.com/pine-script-reference/v6/#var_timenow)[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#var_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.

When used on timeframes higher than 1D, time\_tradingday returns the trading day of the last day inside the bar (e.g. on 1W, it will return the last trading day of the week).

SEE ALSO

[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[time\_close](https://www.tradingview.com/pine-script-reference/v6/#var_time_close)

**timeframe.isdaily**

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

TYPE

simple bool

SEE ALSO

[timeframe.isdwm](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdwm)[timeframe.isintraday](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isintraday)[timeframe.isminutes](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isminutes)[timeframe.isseconds](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isseconds)[timeframe.isticks](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isticks)[timeframe.isweekly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isweekly)[timeframe.ismonthly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.ismonthly)

**timeframe.isdwm**

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

TYPE

simple bool

SEE ALSO

[timeframe.isintraday](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isintraday)[timeframe.isminutes](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isminutes)[timeframe.isseconds](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isseconds)[timeframe.isticks](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isticks)[timeframe.isdaily](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdaily)[timeframe.isweekly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isweekly)[timeframe.ismonthly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.ismonthly)

**timeframe.isintraday**

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

TYPE

simple bool

SEE ALSO

[timeframe.isminutes](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isminutes)[timeframe.isseconds](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isseconds)[timeframe.isticks](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isticks)[timeframe.isdwm](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdwm)[timeframe.isdaily](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdaily)[timeframe.isweekly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isweekly)[timeframe.ismonthly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.ismonthly)

**timeframe.isminutes**

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

TYPE

simple bool

SEE ALSO

[timeframe.isdwm](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdwm)[timeframe.isintraday](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isintraday)[timeframe.isseconds](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isseconds)[timeframe.isticks](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isticks)[timeframe.isdaily](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdaily)[timeframe.isweekly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isweekly)[timeframe.ismonthly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.ismonthly)

**timeframe.ismonthly**

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

TYPE

simple bool

SEE ALSO

[timeframe.isdwm](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdwm)[timeframe.isintraday](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isintraday)[timeframe.isminutes](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isminutes)[timeframe.isseconds](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isseconds)[timeframe.isticks](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isticks)[timeframe.isdaily](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdaily)[timeframe.isweekly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isweekly)

**timeframe.isseconds**

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

TYPE

simple bool

SEE ALSO

[timeframe.isdwm](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdwm)[timeframe.isintraday](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isintraday)[timeframe.isminutes](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isminutes)[timeframe.isticks](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isticks)[timeframe.isdaily](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdaily)[timeframe.isweekly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isweekly)[timeframe.ismonthly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.ismonthly)

**timeframe.isticks**

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

TYPE

simple bool

SEE ALSO

[timeframe.isdwm](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdwm)[timeframe.isintraday](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isintraday)[timeframe.isminutes](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isminutes)[timeframe.isseconds](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isseconds)[timeframe.isdaily](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdaily)[timeframe.isweekly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isweekly)[timeframe.ismonthly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.ismonthly)

**timeframe.isweekly**

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

TYPE

simple bool

SEE ALSO

[timeframe.isdwm](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdwm)[timeframe.isintraday](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isintraday)[timeframe.isminutes](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isminutes)[timeframe.isseconds](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isseconds)[timeframe.isticks](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isticks)[timeframe.isdaily](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.isdaily)[timeframe.ismonthly](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.ismonthly)

**timeframe.main\_period**

A string representation of the script's main timeframe. If the script is an [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period), this variable's value does not change when used in the expression argument of a request.\*() function call.

The string's format is "<quantity>[<unit>]", where <unit> is "T" for ticks, "S" for seconds, "D" for days, "W" for weeks, and "M" for months, but is absent for minutes. No <unit> exists for hours: hourly timeframes are expressed in minutes.

The variable's value is: "10S" for 10 seconds, "30" for 30 minutes, "240" for four hours, "1D" for one day, "2W" for two weeks, and "3M" for one quarter.

TYPE

simple string

SEE ALSO

[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period)[syminfo.main\_tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.main_tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[timeframe.multiplier](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.multiplier)

**timeframe.multiplier**

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

TYPE

simple int

SEE ALSO

[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.main_period)), which equals either the timeframe argument of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) declaration statement or the chart's timeframe.

The string's format is "<quantity>[<unit>]", where <unit> is "T" for ticks, "S" for seconds, "D" for days, "W" for weeks, and "M" for months, but is absent for minutes. No <unit> exists for hours: hourly timeframes are expressed in minutes.

The variable's value is: "10S" for 10 seconds, "30" for 30 minutes, "240" for four hours, "1D" for one day, "2W" for two weeks, and "3M" for one quarter.

TYPE

simple string

REMARKS

To always access the script's main timeframe, even within another context, use the [timeframe.main\_period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.main_period) variable.

SEE ALSO

[timeframe.main\_period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.main_period)[syminfo.main\_tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.main_tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[timeframe.multiplier](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[timestamp](https://www.tradingview.com/pine-script-reference/v6/#fun_timestamp)[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[time\_close](https://www.tradingview.com/pine-script-reference/v6/#var_time_close)[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#var_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

[open](https://www.tradingview.com/pine-script-reference/v6/#var_open)[high](https://www.tradingview.com/pine-script-reference/v6/#var_high)[low](https://www.tradingview.com/pine-script-reference/v6/#var_low)[close](https://www.tradingview.com/pine-script-reference/v6/#var_close)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[hl2](https://www.tradingview.com/pine-script-reference/v6/#var_hl2)[hlc3](https://www.tradingview.com/pine-script-reference/v6/#var_hlc3)[hlcc4](https://www.tradingview.com/pine-script-reference/v6/#var_hlcc4)[ohlc4](https://www.tradingview.com/pine-script-reference/v6/#var_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

[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#fun_weekofyear)[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#var_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

[year](https://www.tradingview.com/pine-script-reference/v6/#fun_year)[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#var_second)

## **Constants**

**adjustment.dividends**

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

TYPE

const string

SEE ALSO

[adjustment.none](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.none)[adjustment.splits](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.splits)[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)

**adjustment.none**

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

TYPE

const string

SEE ALSO

[adjustment.splits](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.splits)[adjustment.dividends](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.dividends)[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)

**adjustment.splits**

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

TYPE

const string

SEE ALSO

[adjustment.none](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.none)[adjustment.dividends](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.dividends)[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)

**alert.freq\_all**

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

All function calls trigger the alert.

TYPE

const string

SEE ALSO

[alert](https://www.tradingview.com/pine-script-reference/v6/#fun_alert)

**alert.freq\_once\_per\_bar**

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

The first function call during the bar triggers the alert.

TYPE

const string

SEE ALSO

[alert](https://www.tradingview.com/pine-script-reference/v6/#fun_alert)

**alert.freq\_once\_per\_bar\_close**

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

The function call triggers the alert only when it occurs during the last script iteration of the real-time bar, when it closes.

TYPE

const string

SEE ALSO

[alert](https://www.tradingview.com/pine-script-reference/v6/#fun_alert)

**backadjustment.inherit**

A constant to specify the value of the backadjustment parameter in [ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new) and [ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify) functions.

TYPE

const backadjustment

SEE ALSO

[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify)[backadjustment.on](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.on)[backadjustment.off](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.off)

**backadjustment.off**

A constant to specify the value of the backadjustment parameter in [ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new) and [ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify) functions.

TYPE

const backadjustment

SEE ALSO

[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify)[backadjustment.on](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.on)[backadjustment.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.inherit)

**backadjustment.on**

A constant to specify the value of the backadjustment parameter in [ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new) and [ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify) functions.

TYPE

const backadjustment

SEE ALSO

[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify)[backadjustment.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.inherit)[backadjustment.off](https://www.tradingview.com/pine-script-reference/v6/#const_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

[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on)

**barmerge.gaps\_on**

Merge strategy for requested data. Data is merged with possible gaps ([na](https://www.tradingview.com/pine-script-reference/v6/#var_na) values).

TYPE

const barmerge\_gaps

SEE ALSO

[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_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

[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[barmerge.lookahead\_on](https://www.tradingview.com/pine-script-reference/v6/#const_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

[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[barmerge.lookahead\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_off)

**color.aqua**

Is a named constant for #00BCD4 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.black**

Is a named constant for #363A45 color.

TYPE

const color

SEE ALSO

[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.blue**

Is a named constant for #2962ff color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.fuchsia**

Is a named constant for #E040FB color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.gray**

Is a named constant for #787B86 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.green**

Is a named constant for #4CAF50 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.lime**

Is a named constant for #00E676 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.maroon**

Is a named constant for #880E4F color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.navy**

Is a named constant for #311B92 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.olive**

Is a named constant for #808000 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.orange**

Is a named constant for #FF9800 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)

**color.purple**

Is a named constant for #9C27B0 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.red**

Is a named constant for #F23645 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.silver**

Is a named constant for #B2B5BE color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.teal**

Is a named constant for #089981 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.white**

Is a named constant for #FFFFFF color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.yellow](https://www.tradingview.com/pine-script-reference/v6/#const_color.yellow)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**color.yellow**

Is a named constant for #FDD835 color.

TYPE

const color

SEE ALSO

[color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black)[color.silver](https://www.tradingview.com/pine-script-reference/v6/#const_color.silver)[color.gray](https://www.tradingview.com/pine-script-reference/v6/#const_color.gray)[color.white](https://www.tradingview.com/pine-script-reference/v6/#const_color.white)[color.maroon](https://www.tradingview.com/pine-script-reference/v6/#const_color.maroon)[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red)[color.purple](https://www.tradingview.com/pine-script-reference/v6/#const_color.purple)[color.fuchsia](https://www.tradingview.com/pine-script-reference/v6/#const_color.fuchsia)[color.green](https://www.tradingview.com/pine-script-reference/v6/#const_color.green)[color.lime](https://www.tradingview.com/pine-script-reference/v6/#const_color.lime)[color.olive](https://www.tradingview.com/pine-script-reference/v6/#const_color.olive)[color.navy](https://www.tradingview.com/pine-script-reference/v6/#const_color.navy)[color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue)[color.teal](https://www.tradingview.com/pine-script-reference/v6/#const_color.teal)[color.aqua](https://www.tradingview.com/pine-script-reference/v6/#const_color.aqua)[color.orange](https://www.tradingview.com/pine-script-reference/v6/#const_color.orange)

**currency.AUD**

Australian dollar.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.BTC**

Bitcoin.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.CAD**

Canadian dollar.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.CHF**

Swiss franc.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.ETH**

Ethereum.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.EUR**

Euro.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.GBP**

Pound sterling.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.HKD**

Hong Kong dollar.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.INR**

Indian rupee.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.JPY**

Japanese yen.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.KRW**

South Korean won.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.MYR**

Malaysian ringgit.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.NOK**

Norwegian krone.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.NONE**

Unspecified currency.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.NZD**

New Zealand dollar.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.RUB**

Russian ruble.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.SEK**

Swedish krona.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.SGD**

Singapore dollar.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.TRY**

Turkish lira.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.USD**

United States dollar.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.USDT**

Tether.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**currency.ZAR**

South African rand.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**dayofweek.friday**

Is a named constant for return value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek) function and value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek) variable.

TYPE

const int

SEE ALSO

[dayofweek.sunday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.sunday)[dayofweek.monday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.monday)[dayofweek.tuesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.tuesday)[dayofweek.wednesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.wednesday)[dayofweek.thursday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.thursday)[dayofweek.saturday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.saturday)

**dayofweek.monday**

Is a named constant for return value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek) function and value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek) variable.

TYPE

const int

SEE ALSO

[dayofweek.sunday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.sunday)[dayofweek.tuesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.tuesday)[dayofweek.wednesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.wednesday)[dayofweek.thursday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.thursday)[dayofweek.friday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.friday)[dayofweek.saturday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.saturday)

**dayofweek.saturday**

Is a named constant for return value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek) function and value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek) variable.

TYPE

const int

SEE ALSO

[dayofweek.sunday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.sunday)[dayofweek.monday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.monday)[dayofweek.tuesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.tuesday)[dayofweek.wednesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.wednesday)[dayofweek.thursday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.thursday)[dayofweek.friday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.friday)

**dayofweek.sunday**

Is a named constant for return value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek) function and value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek) variable.

TYPE

const int

SEE ALSO

[dayofweek.monday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.monday)[dayofweek.tuesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.tuesday)[dayofweek.wednesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.wednesday)[dayofweek.thursday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.thursday)[dayofweek.friday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.friday)[dayofweek.saturday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.saturday)

**dayofweek.thursday**

Is a named constant for return value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek) function and value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek) variable.

TYPE

const int

SEE ALSO

[dayofweek.sunday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.sunday)[dayofweek.monday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.monday)[dayofweek.tuesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.tuesday)[dayofweek.wednesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.wednesday)[dayofweek.friday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.friday)[dayofweek.saturday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.saturday)

**dayofweek.tuesday**

Is a named constant for return value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek) function and value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek) variable.

TYPE

const int

SEE ALSO

[dayofweek.sunday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.sunday)[dayofweek.monday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.monday)[dayofweek.wednesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.wednesday)[dayofweek.thursday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.thursday)[dayofweek.friday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.friday)[dayofweek.saturday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.saturday)

**dayofweek.wednesday**

Is a named constant for return value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek) function and value of [dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek) variable.

TYPE

const int

SEE ALSO

[dayofweek.sunday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.sunday)[dayofweek.monday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.monday)[dayofweek.tuesday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.tuesday)[dayofweek.thursday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.thursday)[dayofweek.friday](https://www.tradingview.com/pine-script-reference/v6/#const_dayofweek.friday)[dayofweek.saturday](https://www.tradingview.com/pine-script-reference/v6/#const_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

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[plotarrow](https://www.tradingview.com/pine-script-reference/v6/#fun_plotarrow)[plotbar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotbar)[plotcandle](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[plotarrow](https://www.tradingview.com/pine-script-reference/v6/#fun_plotarrow)[plotbar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotbar)[plotcandle](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[plotarrow](https://www.tradingview.com/pine-script-reference/v6/#fun_plotarrow)[plotbar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotbar)[plotcandle](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[plotarrow](https://www.tradingview.com/pine-script-reference/v6/#fun_plotarrow)[plotbar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotbar)[plotcandle](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[plotarrow](https://www.tradingview.com/pine-script-reference/v6/#fun_plotarrow)[plotbar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotbar)[plotcandle](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[plotarrow](https://www.tradingview.com/pine-script-reference/v6/#fun_plotarrow)[plotbar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotbar)[plotcandle](https://www.tradingview.com/pine-script-reference/v6/#fun_plotcandle)

**dividends.gross**

A named constant for the [request.dividends](https://www.tradingview.com/pine-script-reference/v6/#fun_request.dividends) function. Is used to request the dividends return on a stock before deductions.

TYPE

const string

SEE ALSO

[request.dividends](https://www.tradingview.com/pine-script-reference/v6/#fun_request.dividends)

**dividends.net**

A named constant for the [request.dividends](https://www.tradingview.com/pine-script-reference/v6/#fun_request.dividends) function. Is used to request the dividends return on a stock after deductions.

TYPE

const string

SEE ALSO

[request.dividends](https://www.tradingview.com/pine-script-reference/v6/#fun_request.dividends)

**earnings.actual**

A named constant for the [request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings) function. Is used to request the earnings value as it was reported.

TYPE

const string

SEE ALSO

[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings)

**earnings.estimate**

A named constant for the [request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings) function. Is used to request the estimated earnings value.

TYPE

const string

SEE ALSO

[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings)

**earnings.standardized**

A named constant for the [request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings) function. Is used to request the standardized earnings value.

TYPE

const string

SEE ALSO

[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings)

**extend.both**

A named constant for [line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) and [line.set\_extend](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_extend) functions.

TYPE

const string

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.set\_extend](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_extend)[extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none)[extend.left](https://www.tradingview.com/pine-script-reference/v6/#const_extend.left)[extend.right](https://www.tradingview.com/pine-script-reference/v6/#const_extend.right)

**extend.left**

A named constant for [line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) and [line.set\_extend](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_extend) functions.

TYPE

const string

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.set\_extend](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_extend)[extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none)[extend.right](https://www.tradingview.com/pine-script-reference/v6/#const_extend.right)[extend.both](https://www.tradingview.com/pine-script-reference/v6/#const_extend.both)

**extend.none**

A named constant for [line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) and [line.set\_extend](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_extend) functions.

TYPE

const string

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.set\_extend](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_extend)[extend.left](https://www.tradingview.com/pine-script-reference/v6/#const_extend.left)[extend.right](https://www.tradingview.com/pine-script-reference/v6/#const_extend.right)[extend.both](https://www.tradingview.com/pine-script-reference/v6/#const_extend.both)

**extend.right**

A named constant for [line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) and [line.set\_extend](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_extend) functions.

TYPE

const string

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.set\_extend](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_extend)[extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none)[extend.left](https://www.tradingview.com/pine-script-reference/v6/#const_extend.left)[extend.both](https://www.tradingview.com/pine-script-reference/v6/#const_extend.both)

**false**

Literal representing a [bool](https://www.tradingview.com/pine-script-reference/v6/#type_bool) value, and result of a comparison operation.

REMARKS

See the User Manual for [comparison operators](https://www.tradingview.com/pine-script-docs/language/operators/#comparison-operators) and [logical operators](https://www.tradingview.com/pine-script-docs/language/operators/#logical-operators).

SEE ALSO

[bool](https://www.tradingview.com/pine-script-reference/v6/#type_bool)

**font.family\_default**

Default text font for [box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new), [box.set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_font_family), [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new), [label.set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_text_font_family), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) and [table.cell\_set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_font_family) functions.

TYPE

const string

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_font_family)[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_text_font_family)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.cell\_set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_font_family)[font.family\_monospace](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_monospace)

**font.family\_monospace**

Monospace text font for [box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new), [box.set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_font_family), [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new), [label.set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_text_font_family), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) and [table.cell\_set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_font_family) functions.

TYPE

const string

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_font_family)[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_text_font_family)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.cell\_set\_text\_font\_family](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_font_family)[font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) function.

TYPE

const string

SEE ALSO

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)[format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price)[format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume)[format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent)

**format.mintick**

Is a named constant to use with the [str.tostring](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tostring) function. Passing a number to [str.tostring](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tostring) with this argument rounds the number to the nearest value that can be divided by [syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_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

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)[format.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_format.inherit)[format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price)[format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) function.

SEE ALSO

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)[format.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_format.inherit)[format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price)[format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume)

**format.price**

Is a named constant for selecting the formatting of the script output values as prices in the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)[format.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_format.inherit)[format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume)[format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent)

**format.volume**

Is a named constant for selecting the formatting of the script output values as volume in the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) function, e.g. '5183' will be formatted as '5.183K'.

The decimal precision rules defined by this variable take precedence over other precision settings. When an [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator), [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy), or plot\*() call uses this format option, the function's precision parameter will not affect the result.

TYPE

const string

SEE ALSO

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)[format.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_format.inherit)[format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price)[format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent)

**hline.style\_dashed**

Is a named constant for dashed linestyle of [hline](https://www.tradingview.com/pine-script-reference/v6/#fun_hline) function.

TYPE

const hline\_style

SEE ALSO

[hline.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_hline.style_solid)[hline.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_hline.style_dotted)

**hline.style\_dotted**

Is a named constant for dotted linestyle of [hline](https://www.tradingview.com/pine-script-reference/v6/#fun_hline) function.

TYPE

const hline\_style

SEE ALSO

[hline.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_hline.style_solid)[hline.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_hline.style_dashed)

**hline.style\_solid**

Is a named constant for solid linestyle of [hline](https://www.tradingview.com/pine-script-reference/v6/#fun_hline) function.

TYPE

const hline\_style

SEE ALSO

[hline.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_hline.style_dotted)[hline.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_hline.style_dashed)

**label.style\_arrowdown**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_arrowup**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_circle**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_cross**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_diamond**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)

**label.style\_flag**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_label\_center**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_label\_down**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_label\_left**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_label\_lower\_left**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_label\_lower\_right**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_label\_right**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_label\_up**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_label\_upper\_left**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_label\_upper\_right**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_none**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_square**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_text\_outline**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_triangledown**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_triangleup**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left)[label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right)[label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left)[label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**label.style\_xcross**

Label style for [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none)[label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross)[label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup)[label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown)[label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag)[label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle)[label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup)[label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown)[label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up)[label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down)[label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left)[label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right)[label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center)[label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square)[label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond)

**line.style\_arrow\_both**

Line style for [line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) and [line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style) functions. Solid line with arrows on both points.

TYPE

const string

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style)[line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid)[line.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dotted)[line.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dashed)[line.style\_arrow\_left](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_left)[line.style\_arrow\_right](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_right)

**line.style\_arrow\_left**

Line style for [line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) and [line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style) functions. Solid line with arrow on the first point.

TYPE

const string

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style)[line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid)[line.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dotted)[line.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dashed)[line.style\_arrow\_right](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_right)[line.style\_arrow\_both](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_both)

**line.style\_arrow\_right**

Line style for [line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) and [line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style) functions. Solid line with arrow on the second point.

TYPE

const string

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style)[line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid)[line.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dotted)[line.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dashed)[line.style\_arrow\_left](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_left)[line.style\_arrow\_both](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_both)

**line.style\_dashed**

Line style for [line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) and [line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style) functions.

TYPE

const string

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style)[line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid)[line.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dotted)[line.style\_arrow\_left](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_left)[line.style\_arrow\_right](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_right)[line.style\_arrow\_both](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_both)

**line.style\_dotted**

Line style for [line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) and [line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style) functions.

TYPE

const string

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style)[line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid)[line.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dashed)[line.style\_arrow\_left](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_left)[line.style\_arrow\_right](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_right)[line.style\_arrow\_both](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_both)

**line.style\_solid**

Line style for [line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) and [line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style) functions.

TYPE

const string

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style)[line.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dotted)[line.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dashed)[line.style\_arrow\_left](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_left)[line.style\_arrow\_right](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_right)[line.style\_arrow\_both](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_both)

**location.abovebar**

Location value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. Shape is plotted above main series bars.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[location.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_location.belowbar)[location.top](https://www.tradingview.com/pine-script-reference/v6/#const_location.top)[location.bottom](https://www.tradingview.com/pine-script-reference/v6/#const_location.bottom)[location.absolute](https://www.tradingview.com/pine-script-reference/v6/#const_location.absolute)

**location.absolute**

Location value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. Shape is plotted on chart using indicator value as a price coordinate.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[location.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_location.abovebar)[location.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_location.belowbar)[location.top](https://www.tradingview.com/pine-script-reference/v6/#const_location.top)[location.bottom](https://www.tradingview.com/pine-script-reference/v6/#const_location.bottom)

**location.belowbar**

Location value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. Shape is plotted below main series bars.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[location.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_location.abovebar)[location.top](https://www.tradingview.com/pine-script-reference/v6/#const_location.top)[location.bottom](https://www.tradingview.com/pine-script-reference/v6/#const_location.bottom)[location.absolute](https://www.tradingview.com/pine-script-reference/v6/#const_location.absolute)

**location.bottom**

Location value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. Shape is plotted near the bottom chart border.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[location.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_location.abovebar)[location.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_location.belowbar)[location.top](https://www.tradingview.com/pine-script-reference/v6/#const_location.top)[location.absolute](https://www.tradingview.com/pine-script-reference/v6/#const_location.absolute)

**location.top**

Location value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. Shape is plotted near the top chart border.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[location.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_location.abovebar)[location.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_location.belowbar)[location.bottom](https://www.tradingview.com/pine-script-reference/v6/#const_location.bottom)[location.absolute](https://www.tradingview.com/pine-script-reference/v6/#const_location.absolute)

**math.e**

Is a named constant for [Euler's number](https://en.wikipedia.org/wiki/E_(mathematical_constant)). It is equal to 2.7182818284590452.

TYPE

const float

SEE ALSO

[math.phi](https://www.tradingview.com/pine-script-reference/v6/#const_math.phi)[math.pi](https://www.tradingview.com/pine-script-reference/v6/#const_math.pi)[math.rphi](https://www.tradingview.com/pine-script-reference/v6/#const_math.rphi)

**math.phi**

Is a named constant for the [golden ratio](https://en.wikipedia.org/wiki/Golden_ratio). It is equal to 1.6180339887498948.

TYPE

const float

SEE ALSO

[math.e](https://www.tradingview.com/pine-script-reference/v6/#const_math.e)[math.pi](https://www.tradingview.com/pine-script-reference/v6/#const_math.pi)[math.rphi](https://www.tradingview.com/pine-script-reference/v6/#const_math.rphi)

**math.pi**

Is a named constant for [Archimedes' constant](https://en.wikipedia.org/wiki/Pi). It is equal to 3.1415926535897932.

TYPE

const float

SEE ALSO

[math.e](https://www.tradingview.com/pine-script-reference/v6/#const_math.e)[math.phi](https://www.tradingview.com/pine-script-reference/v6/#const_math.phi)[math.rphi](https://www.tradingview.com/pine-script-reference/v6/#const_math.rphi)

**math.rphi**

Is a named constant for the [golden ratio conjugate](https://en.wikipedia.org/wiki/Golden_ratio#Golden_ratio_conjugate). It is equal to 0.6180339887498948.

TYPE

const float

SEE ALSO

[math.e](https://www.tradingview.com/pine-script-reference/v6/#const_math.e)[math.pi](https://www.tradingview.com/pine-script-reference/v6/#const_math.pi)[math.phi](https://www.tradingview.com/pine-script-reference/v6/#const_math.phi)

**order.ascending**

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

TYPE

const sort\_order

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sort)

**order.descending**

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

TYPE

const sort\_order

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr)[plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line)[plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr)[plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline)[plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond)[plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram)[plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr)[plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross)[plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns)[plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function. Similar to [plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area), except the gaps in the data are not filled.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr)[plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line)[plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr)[plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline)[plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond)[plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram)[plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross)[plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area)[plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns)[plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr)[plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line)[plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr)[plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline)[plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond)[plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram)[plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross)[plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area)[plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr)[plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr)[plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line)[plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr)[plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline)[plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond)[plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram)[plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross)[plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area)[plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr)[plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr)[plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line)[plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr)[plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline)[plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond)[plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram)[plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area)[plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr)[plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns)[plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr)[plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line)[plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr)[plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline)[plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond)[plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross)[plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area)[plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr)[plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns)[plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr)[plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr)[plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline)[plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond)[plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram)[plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross)[plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area)[plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr)[plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns)[plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function. Similar to [plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line), except the gaps in the data are not filled.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr)[plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line)[plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline)[plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond)[plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram)[plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross)[plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area)[plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr)[plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns)[plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line)[plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr)[plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond)[plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr)[plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram)[plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross)[plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area)[plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr)[plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns)[plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function. Similar to [plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline), except the data changes are also marked with the Diamond shapes.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr)[plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line)[plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr)[plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram)[plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross)[plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area)[plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr)[plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns)[plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_plot) function.

TYPE

const plot\_style

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_circles)[plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line)[plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr)[plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline)[plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond)[plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram)[plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross)[plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area)[plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr)[plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns)

**position.bottom\_center**

Table position is used in [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) functions.

Binds the table to the bottom edge in the center.

TYPE

const string

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)[position.top\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_left)[position.top\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_center)[position.top\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_right)[position.middle\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_left)[position.middle\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_center)[position.middle\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_right)[position.bottom\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_left)

**position.bottom\_left**

Table position is used in [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) functions.

Binds the table to the bottom left of the screen.

TYPE

const string

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)[position.top\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_left)[position.top\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_center)[position.top\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_right)[position.middle\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_left)[position.middle\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_center)[position.middle\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_right)[position.bottom\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_center)

**position.bottom\_right**

Table position is used in [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) functions.

Binds the table to the bottom right of the screen.

TYPE

const string

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)[position.top\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_left)[position.top\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_center)[position.top\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_right)[position.middle\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_left)[position.middle\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_center)[position.middle\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_right)[position.bottom\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_left)[position.bottom\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_center)

**position.middle\_center**

Table position is used in [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) functions.

Binds the table to the center of the screen.

TYPE

const string

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)[position.top\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_left)[position.top\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_center)[position.top\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_right)[position.middle\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_left)[position.middle\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_right)[position.bottom\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_left)[position.bottom\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_center)

**position.middle\_left**

Table position is used in [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) functions.

Binds the table to the left side of the screen.

TYPE

const string

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)[position.top\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_left)[position.top\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_center)[position.top\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_right)[position.middle\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_center)[position.middle\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_right)[position.bottom\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_left)[position.bottom\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_center)

**position.middle\_right**

Table position is used in [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) functions.

Binds the table to the right side of the screen.

TYPE

const string

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)[position.top\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_left)[position.top\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_center)[position.top\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_right)[position.middle\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_left)[position.middle\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_center)[position.bottom\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_left)[position.bottom\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_center)

**position.top\_center**

Table position is used in [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) functions.

Binds the table to the top edge in the center.

TYPE

const string

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)[position.top\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_left)[position.top\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_right)[position.middle\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_left)[position.middle\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_center)[position.middle\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_right)[position.bottom\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_left)[position.bottom\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_center)

**position.top\_left**

Table position is used in [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) functions.

Binds the table to the upper-left edge.

TYPE

const string

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)[position.top\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_center)[position.top\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_right)[position.middle\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_left)[position.middle\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_center)[position.middle\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_right)[position.bottom\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_left)[position.bottom\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_center)

**position.top\_right**

Table position is used in [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) functions.

Binds the table to the upper-right edge.

TYPE

const string

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)[position.top\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_left)[position.top\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_center)[position.middle\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_left)[position.middle\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_center)[position.middle\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_right)[position.bottom\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_left)[position.bottom\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_center)

**scale.left**

Scale value for [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) function. Indicator is added to the left price scale.

TYPE

const scale\_type

SEE ALSO

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)

**scale.none**

Scale value for [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) function. Indicator is added in 'No Scale' mode. Can be used only with 'overlay=true'.

TYPE

const scale\_type

SEE ALSO

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)

**scale.right**

Scale value for [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) function. Indicator is added to the right price scale.

TYPE

const scale\_type

SEE ALSO

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)

**session.extended**

Constant for extended session type (with extended hours data).

TYPE

const string

SEE ALSO

[session.regular](https://www.tradingview.com/pine-script-reference/v6/#const_session.regular)[syminfo.session](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.session)

**session.regular**

Constant for regular session type (no extended hours data).

TYPE

const string

SEE ALSO

[session.extended](https://www.tradingview.com/pine-script-reference/v6/#const_session.extended)[syminfo.session](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.session)

**settlement\_as\_close.inherit**

A constant to specify the value of the settlement\_as\_close parameter in [ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new) and [ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify) functions.

TYPE

const settlement

SEE ALSO

[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify)[settlement\_as\_close.on](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.on)[settlement\_as\_close.off](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.off)

**settlement\_as\_close.off**

A constant to specify the value of the settlement\_as\_close parameter in [ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new) and [ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify) functions.

TYPE

const settlement

SEE ALSO

[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify)[settlement\_as\_close.on](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.on)[settlement\_as\_close.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.inherit)

**settlement\_as\_close.on**

A constant to specify the value of the settlement\_as\_close parameter in [ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new) and [ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify) functions.

TYPE

const settlement

SEE ALSO

[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify)[settlement\_as\_close.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.inherit)[settlement\_as\_close.off](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.off)

**shape.arrowdown**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.arrowup**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.circle**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.cross**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.diamond**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.flag**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.labeldown**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.labelup**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.square**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.triangledown**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.triangleup**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**shape.xcross**

Shape style for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)

**size.auto**

Size value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. The size of the shape automatically adapts to the size of the bars.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[label.set\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_size)[size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny)[size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small)[size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal)[size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large)[size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge)

**size.huge**

Size value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. The size of the shape constantly huge.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[label.set\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_size)[size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto)[size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny)[size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small)[size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal)[size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large)

**size.large**

Size value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. The size of the shape constantly large.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[label.set\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_size)[size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto)[size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny)[size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small)[size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal)[size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge)

**size.normal**

Size value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. The size of the shape constantly normal.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[label.set\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_size)[size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto)[size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny)[size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small)[size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large)[size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge)

**size.small**

Size value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. The size of the shape constantly small.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[label.set\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_size)[size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto)[size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny)[size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal)[size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large)[size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge)

**size.tiny**

Size value for [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape), [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) functions. The size of the shape constantly tiny.

TYPE

const string

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[label.set\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_size)[size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto)[size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small)[size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal)[size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large)[size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge)

**splits.denominator**

A named constant for the [request.splits](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[request.splits](https://www.tradingview.com/pine-script-reference/v6/#fun_request.splits)

**splits.numerator**

A named constant for the [request.splits](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[request.splits](https://www.tradingview.com/pine-script-reference/v6/#fun_request.splits)

**strategy.cash**

This is one of the arguments that can be supplied to the default\_qty\_type parameter in the [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) declaration statement. It is only relevant when no value is used for the ‘qty’ parameter in [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) or [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_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

//**@version=**6

strategy("strategy.cash", overlay = true, default\_qty\_value = 50, default\_qty\_type = strategy.cash, initial\_capital = 1000000)

if bar\_index == 0

// As ‘qty’ is not defined, the previously defined values for the `default\_qty\_type` and `default\_qty\_value` parameters are used to enter trades, namely 50 units of cash in the currency of `strategy.account\_currency`.

// `qty` is calculated as (default\_qty\_value)/(close price). If current price is $5, then qty = 50/5 = 10.

strategy.entry("EN", strategy.long)

if bar\_index == 2

strategy.close("EN")

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**strategy.commission.cash\_per\_contract**

Commission type for an order. Money displayed in the account currency per contract.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**strategy.commission.cash\_per\_order**

Commission type for an order. Money displayed in the account currency per order.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**strategy.commission.percent**

Commission type for an order. A percentage of the cash volume of order.

TYPE

const string

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**strategy.direction.all**

It allows strategy to open both long and short positions.

TYPE

const string

SEE ALSO

[strategy.risk.allow\_entry\_in](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.risk.allow_entry_in)

**strategy.direction.long**

It allows strategy to open only long positions.

TYPE

const string

SEE ALSO

[strategy.risk.allow\_entry\_in](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.risk.allow_entry_in)

**strategy.direction.short**

It allows strategy to open only short positions.

TYPE

const string

SEE ALSO

[strategy.risk.allow\_entry\_in](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) declaration statement. It is only relevant when no value is used for the ‘qty’ parameter in [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) or [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order) function calls. It specifies that a number of contracts/shares/lots will be used to enter trades.

TYPE

const string

EXAMPLE

//**@version=**6

strategy("strategy.fixed", overlay = true, default\_qty\_value = 50, default\_qty\_type = strategy.fixed, initial\_capital = 1000000)

if bar\_index == 0

// As ‘qty’ is not defined, the previously defined values for the `default\_qty\_type` and `default\_qty\_value` parameters are used to enter trades, namely 50 contracts.

// qty = 50

strategy.entry("EN", strategy.long)

if bar\_index == 2

strategy.close("EN")

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**strategy.long**

A named constant for use with the direction parameter of the [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) and [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order) commands. It specifies that the command creates a buy order.

TYPE

const strategy\_direction

SEE ALSO

[strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry)[strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit)[strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order)

**strategy.oca.cancel**

A named constant for use with the oca\_type parameter of the [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) and [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order) calls with the same oca\_\* arguments, the strategy cannot fully or partially cancel either one.

SEE ALSO

[strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry)[strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit)[strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order)

**strategy.oca.none**

A named constant for use with the oca\_type parameter of the [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) and [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry)[strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit)[strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order)

**strategy.oca.reduce**

A named constant for use with the oca\_type parameter of the [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) and [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order) calls with the same oca\_\* arguments, the strategy cannot fully or partially cancel either one.

Orders from [strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit) automatically use this OCA type, and they belong to the same OCA group by default.

SEE ALSO

[strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry)[strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit)[strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) declaration statement. It is only relevant when no value is used for the ‘qty’ parameter in [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) or [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order) function calls. It specifies that a percentage (0-100) of equity will be used to enter trades.

TYPE

const string

EXAMPLE

//**@version=**6

strategy("strategy.percent\_of\_equity", overlay = false, default\_qty\_value = 100, default\_qty\_type = strategy.percent\_of\_equity, initial\_capital = 1000000)

// As ‘qty’ is not defined, the previously defined values for the `default\_qty\_type` and `default\_qty\_value` parameters are used to enter trades, namely 100% of available equity.

if bar\_index == 0

strategy.entry("EN", strategy.long)

if bar\_index == 2

strategy.close("EN")

plot(strategy.equity)

// The ‘qty’ parameter is set to 10. Entering position with fixed size of 10 contracts and entry market price = (10 \* close).

if bar\_index == 4

strategy.entry("EN", strategy.long, qty = 10)

if bar\_index == 6

strategy.close("EN")

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**strategy.short**

A named constant for use with the direction parameter of the [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) and [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order) commands. It specifies that the command creates a sell order.

TYPE

const strategy\_direction

SEE ALSO

[strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry)[strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit)[strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order)

**text.align\_bottom**

Vertical text alignment for [box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new), [box.set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_valign), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) and [table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign) functions.

TYPE

const string

SEE ALSO

[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center)[text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left)[text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_right)

**text.align\_center**

Text alignment for [box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new), [box.set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_halign), [box.set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_valign), [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left)[text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_right)

**text.align\_left**

Horizontal text alignment for [box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new), [box.set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_halign), [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center)[text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_right)

**text.align\_right**

Horizontal text alignment for [box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new), [box.set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_halign), [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) and [label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign) functions.

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center)[text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left)

**text.align\_top**

Vertical text alignment for [box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new), [box.set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_valign), [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) and [table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign) functions.

TYPE

const string

SEE ALSO

[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center)[text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left)[text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_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

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)

**text.wrap\_auto**

Automatic wrapping mode for [box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new) and [box.set\_text\_wrap](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_wrap) functions.

TYPE

const string

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text)[box.set\_text\_wrap](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_wrap)

**text.wrap\_none**

Disabled wrapping mode for [box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new) and [box.set\_text\_wrap](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_wrap) functions.

TYPE

const string

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text)[box.set\_text\_wrap](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_wrap)

**true**

Literal representing one of the values a [bool](https://www.tradingview.com/pine-script-reference/v6/#type_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](https://www.tradingview.com/pine-script-docs/language/operators/#comparison-operators) and [logical operators](https://www.tradingview.com/pine-script-docs/language/operators/#logical-operators).

SEE ALSO

[bool](https://www.tradingview.com/pine-script-reference/v6/#type_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

[xloc.bar\_time](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_time)[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[polyline.new](https://www.tradingview.com/pine-script-reference/v6/#fun_polyline.new)[line.set\_xloc](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_xloc)[label.set\_xloc](https://www.tradingview.com/pine-script-reference/v6/#fun_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

[xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index)[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[polyline.new](https://www.tradingview.com/pine-script-reference/v6/#fun_polyline.new)[line.set\_xloc](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_xloc)[label.set\_xloc](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_xloc)[xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index)

**yloc.abovebar**

A named constant that specifies the algorithm of interpretation of y-value in function [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new).

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_yloc](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_yloc)[yloc.price](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.price)[yloc.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.belowbar)

**yloc.belowbar**

A named constant that specifies the algorithm of interpretation of y-value in function [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new).

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_yloc](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_yloc)[yloc.price](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.price)[yloc.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.abovebar)

**yloc.price**

A named constant that specifies the algorithm of interpretation of y-value in function [label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new).

TYPE

const string

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_yloc](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_yloc)[yloc.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.abovebar)[yloc.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.belowbar)

## **Functions**

**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

alert(message, freq) → void

ARGUMENTS

**message (series string)** The message to send when the alert occurs.

**freq (input string)** Optional. Determines the allowed frequency of the alert trigger. Possible values are: [alert.freq\_all](https://www.tradingview.com/pine-script-reference/v6/#const_alert.freq_all) (allows an alert on any realtime update), [alert.freq\_once\_per\_bar](https://www.tradingview.com/pine-script-reference/v6/#const_alert.freq_once_per_bar) (allows an alert only on the first execution for each realtime bar), or [alert.freq\_once\_per\_bar\_close](https://www.tradingview.com/pine-script-reference/v6/#const_alert.freq_once_per_bar_close) (allows an alert only when a realtime bar closes). The default is [alert.freq\_once\_per\_bar](https://www.tradingview.com/pine-script-reference/v6/#const_alert.freq_once_per_bar).

EXAMPLE

//**@version=**6

indicator("`alert()` example", "", true)

ma = ta.sma(close, 14)

xUp = ta.crossover(close, ma)

if xUp

// Trigger the alert the first time a cross occurs during the real-time bar.

alert("Price (" + str.tostring(close) + ") crossed over MA (" + str.tostring(ma) + ").", alert.freq\_once\_per\_bar)

plot(ma)

plotchar(xUp, "xUp", "▲", location.top, size = size.tiny)

REMARKS

The alert() function does not display information on the chart.

In contrast to [alertcondition](https://www.tradingview.com/pine-script-reference/v6/#fun_alertcondition), calls to this function do not count toward a script's plot count. Additionally, alert() calls are allowed in local scopes, including the scopes of exported library functions.

See [this article](https://www.tradingview.com/chart/?solution=43000597494) in our Help Center to learn more about activating alerts from alert() calls.

SEE ALSO

[alertcondition](https://www.tradingview.com/pine-script-reference/v6/#fun_alertcondition)

**alertcondition()**

Creates alert condition, that is available in Create Alert dialog. Please note, that [alertcondition](https://www.tradingview.com/pine-script-reference/v6/#fun_alertcondition) does NOT create an alert, it just gives you more options in Create Alert dialog. Also, [alertcondition](https://www.tradingview.com/pine-script-reference/v6/#fun_alertcondition) effect is invisible on chart.

SYNTAX

alertcondition(condition, title, message) → void

ARGUMENTS

**condition (series bool)** Series of boolean values that is used for alert. True values mean alert fire, false - no alert. Required argument.

**title (const string)** Title of the alert condition. Optional argument.

**message (const string)** Message to display when alert fires. Optional argument.

EXAMPLE

//**@version=**6

indicator("alertcondition", overlay=true)

alertcondition(close >= open, title='Alert on Green Bar', message='Green Bar!')

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

[alert](https://www.tradingview.com/pine-script-reference/v6/#fun_alert)

**array.abs()**2 overloads

Returns an array containing the absolute value of each element in the original array.

SYNTAX & OVERLOADS

[array.abs(id) → array<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.abs-0)

[array.abs(id) → array<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.abs-1)

ARGUMENTS

**id (array<int/float>)** An array object.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.reverse](https://www.tradingview.com/pine-script-reference/v6/#fun_array.reverse)[order.ascending](https://www.tradingview.com/pine-script-reference/v6/#const_order.ascending)[order.descending](https://www.tradingview.com/pine-script-reference/v6/#const_order.descending)

**array.avg()**2 overloads

The function returns the mean of an array's elements.

SYNTAX & OVERLOADS

[array.avg(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.avg-0)

[array.avg(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.avg-1)

ARGUMENTS

**id (array<int/float>)** An array object.

EXAMPLE

//**@version=**6

indicator("array.avg example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

plot(array.avg(a))

RETURNS

Mean of array's elements.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.max](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max)[array.min](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min)[array.stdev](https://www.tradingview.com/pine-script-reference/v6/#fun_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

array.binary\_search(id, val) → series int

ARGUMENTS

**id (array<int/float>)** An array object.

**val (series int/float)** The value to search for in the array.

EXAMPLE

//**@version=**6

indicator("array.binary\_search")

a = array.from(5, -2, 0, 9, 1)

array.sort(a) // [-2, 0, 1, 5, 9]

position = array.binary\_search(a, 0) // 1

plot(position)

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.reverse](https://www.tradingview.com/pine-script-reference/v6/#fun_array.reverse)[order.ascending](https://www.tradingview.com/pine-script-reference/v6/#const_order.ascending)[order.descending](https://www.tradingview.com/pine-script-reference/v6/#const_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

array.binary\_search\_leftmost(id, val) → series int

ARGUMENTS

**id (array<int/float>)** An array object.

**val (series int/float)** The value to search for in the array.

EXAMPLE

//**@version=**6

indicator("array.binary\_search\_leftmost")

a = array.from(5, -2, 0, 9, 1)

array.sort(a) // [-2, 0, 1, 5, 9]

position = array.binary\_search\_leftmost(a, 3) // 2

plot(position)

EXAMPLE

//**@version=**6

indicator("array.binary\_search\_leftmost, repetitive elements")

a = array.from(4, 5, 5, 5)

// Returns the index of the first instance.

position = array.binary\_search\_leftmost(a, 5)

plot(position) // Plots 1

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.reverse](https://www.tradingview.com/pine-script-reference/v6/#fun_array.reverse)[order.ascending](https://www.tradingview.com/pine-script-reference/v6/#const_order.ascending)[order.descending](https://www.tradingview.com/pine-script-reference/v6/#const_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

array.binary\_search\_rightmost(id, val) → series int

ARGUMENTS

**id (array<int/float>)** An array object.

**val (series int/float)** The value to search for in the array.

EXAMPLE

//**@version=**6

indicator("array.binary\_search\_rightmost")

a = array.from(5, -2, 0, 9, 1)

array.sort(a) // [-2, 0, 1, 5, 9]

position = array.binary\_search\_rightmost(a, 3) // 3

plot(position)

EXAMPLE

//**@version=**6

indicator("array.binary\_search\_rightmost, repetitive elements")

a = array.from(4, 5, 5, 5)

// Returns the index of the last instance.

position = array.binary\_search\_rightmost(a, 5)

plot(position) // Plots 3

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.reverse](https://www.tradingview.com/pine-script-reference/v6/#fun_array.reverse)[order.ascending](https://www.tradingview.com/pine-script-reference/v6/#const_order.ascending)[order.descending](https://www.tradingview.com/pine-script-reference/v6/#const_order.descending)

**array.clear()**

The function removes all elements from an array.

SYNTAX

array.clear(id) → void

ARGUMENTS

**id (any array type)** An array object.

EXAMPLE

//**@version=**6

indicator("array.clear example")

a = array.new\_float(5,high)

array.clear(a)

array.push(a, close)

plot(array.get(a,0))

plot(array.size(a))

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.push](https://www.tradingview.com/pine-script-reference/v6/#fun_array.push)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.pop](https://www.tradingview.com/pine-script-reference/v6/#fun_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

array.concat(id1, id2) → array<type>

ARGUMENTS

**id1 (any array type)** The first array object.

**id2 (any array type)** The second array object.

EXAMPLE

//**@version=**6

indicator("array.concat example")

a = array.new\_float(0,0)

b = array.new\_float(0,0)

for i = 0 to 4

array.push(a, high[i])

array.push(b, low[i])

c = array.concat(a,b)

plot(array.size(a))

plot(array.size(b))

plot(array.size(c))

RETURNS

The first array with merged elements from the second array.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)

**array.copy()**

The function creates a copy of an existing array.

SYNTAX

array.copy(id) → array<type>

ARGUMENTS

**id (any array type)** An array object.

EXAMPLE

//**@version=**6

indicator("array.copy example")

length = 5

a = array.new\_float(length, close)

b = array.copy(a)

a := array.new\_float(length, open)

plot(array.sum(a) / length)

plot(array.sum(b) / length)

RETURNS

A copy of an array.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sort)

**array.covariance()**

The function returns the covariance of two arrays.

SYNTAX

array.covariance(id1, id2, biased) → series float

ARGUMENTS

**id1 (array<int/float>)** An array object.

**id2 (array<int/float>)** An array object.

**biased (series bool)** Determines which estimate should be used. Optional. The default is true.

EXAMPLE

//**@version=**6

indicator("array.covariance example")

a = array.new\_float(0)

b = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

array.push(b, open[i])

plot(array.covariance(a, b))

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.max](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max)[array.stdev](https://www.tradingview.com/pine-script-reference/v6/#fun_array.stdev)[array.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_array.avg)[array.variance](https://www.tradingview.com/pine-script-reference/v6/#fun_array.variance)

**array.every()**

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if all elements of the id array are [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SYNTAX

array.every(id) → series bool

ARGUMENTS

**id (array<bool>)** An array object.

REMARKS

This function also works with arrays of [int](https://www.tradingview.com/pine-script-reference/v6/#type_int) and [float](https://www.tradingview.com/pine-script-reference/v6/#type_float) types, in which case zero values are considered [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), and all others [true](https://www.tradingview.com/pine-script-reference/v6/#const_true).

SEE ALSO

[array.some](https://www.tradingview.com/pine-script-reference/v6/#fun_array.some)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_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

array.fill(id, value, index\_from, index\_to) → void

ARGUMENTS

**id (any array type)** An array object.

**value (series <type of the array's elements>)** Value to fill the array with.

**index\_from (series int)** Start index, default is 0.

**index\_to (series int)** End index, default is na. Must be one greater than the index of the last element to set.

EXAMPLE

//**@version=**6

indicator("array.fill example")

a = array.new\_float(10)

array.fill(a, close)

plot(array.sum(a))

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)

**array.first()**

Returns the array's first element. Throws a runtime error if the array is empty.

SYNTAX

array.first(id) → series <type>

ARGUMENTS

**id (any array type)** An array object.

EXAMPLE

//**@version=**6

indicator("array.first example")

arr = array.new\_int(3, 10)

plot(array.first(arr))

SEE ALSO

[array.last](https://www.tradingview.com/pine-script-reference/v6/#fun_array.last)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)

**array.from()**12 overloads

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

[array.from(arg0, arg1, ...) → array<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-0)

[array.from(arg0, arg1, ...) → array<series enum>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-1)

[array.from(arg0, arg1, ...) → array<label>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-2)

[array.from(arg0, arg1, ...) → array<line>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-3)

[array.from(arg0, arg1, ...) → array<box>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-4)

[array.from(arg0, arg1, ...) → array<table>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-5)

[array.from(arg0, arg1, ...) → array<linefill>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-6)

[array.from(arg0, arg1, ...) → array<string>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-7)

[array.from(arg0, arg1, ...) → array<color>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-8)

[array.from(arg0, arg1, ...) → array<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-9)

[array.from(arg0, arg1, ...) → array<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-10)

[array.from(arg0, arg1, ...) → array<bool>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from-11)

ARGUMENTS

**arg0, arg1, ... (<arg...\_type>)** Array arguments.

EXAMPLE

//**@version=**6

indicator("array.from\_example", overlay = false)

arr = array.from("Hello", "World!") // arr (array<string>) will contain 2 elements: {Hello}, {World!}.

plot(close)

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

array.get(id, index) → series <type>

ARGUMENTS

**id (any array type)** An array object.

**index (series int)** The index of the element whose value is to be returned.

EXAMPLE

//**@version=**6

indicator("array.get example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i] - open[i])

plot(array.get(a, 9))

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sort)

**array.includes()**

The function returns true if the value was found in an array, false otherwise.

SYNTAX

array.includes(id, value) → series bool

ARGUMENTS

**id (any array type)** An array object.

**value (series <type of the array's elements>)** The value to search in the array.

EXAMPLE

//**@version=**6

indicator("array.includes example")

a = array.new\_float(5,high)

p = close

if array.includes(a, high)

p := open

plot(p)

RETURNS

True if the value was found in the array, false otherwise.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.indexof](https://www.tradingview.com/pine-script-reference/v6/#fun_array.indexof)[array.shift](https://www.tradingview.com/pine-script-reference/v6/#fun_array.shift)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_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

array.indexof(id, value) → series int

ARGUMENTS

**id (any array type)** An array object.

**value (series <type of the array's elements>)** The value to search in the array.

EXAMPLE

//**@version=**6

indicator("array.indexof example")

a = array.new\_float(5,high)

index = array.indexof(a, high)

plot(index)

RETURNS

The index of an element.

SEE ALSO

[array.lastindexof](https://www.tradingview.com/pine-script-reference/v6/#fun_array.lastindexof)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.lastindexof](https://www.tradingview.com/pine-script-reference/v6/#fun_array.lastindexof)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)

**array.insert()**

The function changes the contents of an array by adding new elements in place.

SYNTAX

array.insert(id, index, value) → void

ARGUMENTS

**id (any array type)** An array object.

**index (series int)** The index at which to insert the value.

**value (series <type of the array's elements>)** The value to add to the array.

EXAMPLE

//**@version=**6

indicator("array.insert example")

a = array.new\_float(5, close)

array.insert(a, 0, open)

plot(array.get(a, 5))

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.push](https://www.tradingview.com/pine-script-reference/v6/#fun_array.push)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.pop](https://www.tradingview.com/pine-script-reference/v6/#fun_array.pop)[array.unshift](https://www.tradingview.com/pine-script-reference/v6/#fun_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

array.join(id, separator) → series string

ARGUMENTS

**id (array<int/float/string>)** An array object.

**separator (series string)** The string used to separate each array element.

EXAMPLE

//**@version=**6

indicator("array.join example")

a = array.new\_float(5, 5)

label.new(bar\_index, close, array.join(a, ","))

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.pop](https://www.tradingview.com/pine-script-reference/v6/#fun_array.pop)[array.unshift](https://www.tradingview.com/pine-script-reference/v6/#fun_array.unshift)

**array.last()**

Returns the array's last element. Throws a runtime error if the array is empty.

SYNTAX

array.last(id) → series <type>

ARGUMENTS

**id (any array type)** An array object.

EXAMPLE

//**@version=**6

indicator("array.last example")

arr = array.new\_int(3, 10)

plot(array.last(arr))

SEE ALSO

[array.first](https://www.tradingview.com/pine-script-reference/v6/#fun_array.first)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_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

array.lastindexof(id, value) → series int

ARGUMENTS

**id (any array type)** An array object.

**value (series <type of the array's elements>)** The value to search in the array.

EXAMPLE

//**@version=**6

indicator("array.lastindexof example")

a = array.new\_float(5,high)

index = array.lastindexof(a, high)

plot(index)

RETURNS

The index of an element.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.push](https://www.tradingview.com/pine-script-reference/v6/#fun_array.push)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)

**array.max()**4 overloads

The function returns the greatest value, or the nth greatest value in a given array.

SYNTAX & OVERLOADS

[array.max(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max-0)

[array.max(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max-1)

[array.max(id, nth) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max-2)

[array.max(id, nth) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max-3)

ARGUMENTS

**id (array<int/float>)** An array object.

EXAMPLE

//**@version=**6

indicator("array.max")

a = array.from(5, -2, 0, 9, 1)

thirdHighest = array.max(a, 2) // 1

plot(thirdHighest)

RETURNS

The greatest or the nth greatest value in the array.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.min](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min)[array.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sum)

**array.median()**2 overloads

The function returns the median of an array's elements.

SYNTAX & OVERLOADS

[array.median(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.median-0)

[array.median(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.median-1)

ARGUMENTS

**id (array<int/float>)** An array object.

EXAMPLE

//**@version=**6

indicator("array.median example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

plot(array.median(a))

RETURNS

The median of the array's elements.

SEE ALSO

[array.median](https://www.tradingview.com/pine-script-reference/v6/#fun_array.median)[array.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_array.avg)[array.variance](https://www.tradingview.com/pine-script-reference/v6/#fun_array.variance)[array.min](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min)

**array.min()**4 overloads

The function returns the smallest value, or the nth smallest value in a given array.

SYNTAX & OVERLOADS

[array.min(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min-0)

[array.min(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min-1)

[array.min(id, nth) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min-2)

[array.min(id, nth) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min-3)

ARGUMENTS

**id (array<int/float>)** An array object.

EXAMPLE

//**@version=**6

indicator("array.min")

a = array.from(5, -2, 0, 9, 1)

secondLowest = array.min(a, 1) // 0

plot(secondLowest)

RETURNS

The smallest or the nth smallest value in the array.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.max](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max)[array.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sum)

**array.mode()**2 overloads

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

[array.mode(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.mode-0)

[array.mode(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.mode-1)

ARGUMENTS

**id (array<int/float>)** An array object.

EXAMPLE

//**@version=**6

indicator("array.mode example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

plot(array.mode(a))

RETURNS

The most frequently occurring value from the id array. If none exists, returns the smallest value instead.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[ta.mode](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.mode)[matrix.mode](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.mode)[array.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_array.avg)[array.variance](https://www.tradingview.com/pine-script-reference/v6/#fun_array.variance)[array.min](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min)

**array.new\_bool()**

The function creates a new array object of bool type elements.

SYNTAX

array.new\_bool(size, initial\_value) → array<bool>

ARGUMENTS

**size (series int)** Initial size of an array. Optional. The default is 0.

**initial\_value (series bool)** Initial value of all array elements. Optional. The default is 'false'.

EXAMPLE

//**@version=**6

indicator("array.new\_bool example")

length = 5

a = array.new\_bool(length, close > open)

plot(array.get(a, 0) ? close : open)

RETURNS

The ID of an array object which may be used in other array.\*() functions.

REMARKS

An array index starts from 0.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sort)

**array.new\_box()**

The function creates a new array object of box type elements.

SYNTAX

array.new\_box(size, initial\_value) → array<box>

ARGUMENTS

**size (series int)** Initial size of an array. Optional. The default is 0.

**initial\_value (series box)** Initial value of all array elements. Optional. The default is 'na'.

EXAMPLE

//**@version=**6

indicator("array.new\_box example")

boxes = array.new\_box()

array.push(boxes, box.new(time, close, time+2, low, xloc=xloc.bar\_time))

plot(1)

RETURNS

The ID of an array object which may be used in other array.\*() functions.

REMARKS

An array index starts from 0.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)

**array.new\_color()**

The function creates a new array object of color type elements.

SYNTAX

array.new\_color(size, initial\_value) → array<color>

ARGUMENTS

**size (series int)** Initial size of an array. Optional. The default is 0.

**initial\_value (series color)** Initial value of all array elements. Optional. The default is 'na'.

EXAMPLE

//**@version=**6

indicator("array.new\_color example")

length = 5

a = array.new\_color(length, color.red)

plot(close, color = array.get(a, 0))

RETURNS

The ID of an array object which may be used in other array.\*() functions.

REMARKS

An array index starts from 0.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sort)

**array.new\_float()**

The function creates a new array object of float type elements.

SYNTAX

array.new\_float(size, initial\_value) → array<float>

ARGUMENTS

**size (series int)** Initial size of an array. Optional. The default is 0.

**initial\_value (series int/float)** Initial value of all array elements. Optional. The default is 'na'.

EXAMPLE

//**@version=**6

indicator("array.new\_float example")

length = 5

a = array.new\_float(length, close)

plot(array.sum(a) / length)

RETURNS

The ID of an array object which may be used in other array.\*() functions.

REMARKS

An array index starts from 0.

SEE ALSO

[array.new\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_color)[array.new\_bool](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_bool)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sort)

**array.new\_int()**

The function creates a new array object of int type elements.

SYNTAX

array.new\_int(size, initial\_value) → array<int>

ARGUMENTS

**size (series int)** Initial size of an array. Optional. The default is 0.

**initial\_value (series int)** Initial value of all array elements. Optional. The default is 'na'.

EXAMPLE

//**@version=**6

indicator("array.new\_int example")

length = 5

a = array.new\_int(length, int(close))

plot(array.sum(a) / length)

RETURNS

The ID of an array object which may be used in other array.\*() functions.

REMARKS

An array index starts from 0.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sort)

**array.new\_label()**

The function creates a new array object of label type elements.

SYNTAX

array.new\_label(size, initial\_value) → array<label>

ARGUMENTS

**size (series int)** Initial size of an array. Optional. The default is 0.

**initial\_value (series label)** Initial value of all array elements. Optional. The default is 'na'.

EXAMPLE

//**@version=**6

indicator("array.new\_label example", overlay = true, max\_labels\_count = 500)

//**@variable** The number of labels to show on the chart.

**int** labelCount = input.int(50, "Labels to show", 1, 500)

//**@variable** An array of `label` objects.

var **array**<**label**> labelArray = array.new\_label()

//**@variable** A `chart.point` for the new label.

labelPoint = chart.point.from\_index(bar\_index, close)

//**@variable** The text in the new label.

**string** labelText = na

//**@variable** The color of the new label.

**color** labelColor = na

//**@variable** The style of the new label.

**string** labelStyle = na

// Set the label attributes for rising bars.

if close > open

labelText := "Rising"

labelColor := color.green

labelStyle := label.style\_label\_down

// Set the label attributes for falling bars.

else if close < open

labelText := "Falling"

labelColor := color.red

labelStyle := label.style\_label\_up

// Add a new label to the `labelArray` when the chart bar closed at a new value.

if close != open

labelArray.push(label.new(labelPoint, labelText, color = labelColor, style = labelStyle))

// Remove the first element and delete its label when the size of the `labelArray` exceeds the `labelCount`.

if labelArray.size() > labelCount

label.delete(labelArray.shift())

RETURNS

The ID of an array object which may be used in other array.\*() functions.

REMARKS

An array index starts from 0.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)

**array.new\_line()**

The function creates a new array object of line type elements.

SYNTAX

array.new\_line(size, initial\_value) → array<line>

ARGUMENTS

**size (series int)** Initial size of an array. Optional. The default is 0.

**initial\_value (series line)** Initial value of all array elements. Optional. The default is 'na'.

EXAMPLE

//**@version=**6

indicator("array.new\_line example")

// draw last 15 lines

var a = array.new\_line()

array.push(a, line.new(bar\_index - 1, close[1], bar\_index, close))

if array.size(a) > 15

ln = array.shift(a)

line.delete(ln)

RETURNS

The ID of an array object which may be used in other array.\*() functions.

REMARKS

An array index starts from 0.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)

**array.new\_linefill()**

The function creates a new array object of linefill type elements.

SYNTAX

array.new\_linefill(size, initial\_value) → array<linefill>

ARGUMENTS

**size (series int)** Initial size of an array.

**initial\_value (series linefill)** Initial value of all array elements.

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

array.new\_string(size, initial\_value) → array<string>

ARGUMENTS

**size (series int)** Initial size of an array. Optional. The default is 0.

**initial\_value (series string)** Initial value of all array elements. Optional. The default is 'na'.

EXAMPLE

//**@version=**6

indicator("array.new\_string example")

length = 5

a = array.new\_string(length, "text")

label.new(bar\_index, close, array.get(a, 0))

RETURNS

The ID of an array object which may be used in other array.\*() functions.

REMARKS

An array index starts from 0.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)

**array.new\_table()**

The function creates a new array object of table type elements.

SYNTAX

array.new\_table(size, initial\_value) → array<table>

ARGUMENTS

**size (series int)** Initial size of an array. Optional. The default is 0.

**initial\_value (series table)** Initial value of all array elements. Optional. The default is 'na'.

EXAMPLE

//**@version=**6

indicator("table array")

tables = array.new\_table()

array.push(tables, table.new(position = position.top\_left, rows = 1, columns = 2, bgcolor = color.yellow, border\_width=1))

plot(1)

RETURNS

The ID of an array object which may be used in other array.\*() functions.

REMARKS

An array index starts from 0.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)

**array.new<type>()**

The function creates a new array object of <type> elements.

SYNTAX

array.new<type>(size, initial\_value) → array<type>

ARGUMENTS

**size (series int)** Initial size of an array. Optional. The default is 0.

**initial\_value (<array\_type>)** Initial value of all array elements. Optional. The default is 'na'.

EXAMPLE

//**@version=**6

indicator("array.new<string> example")

a = array.new<**string**>(1, "Hello, World!")

label.new(bar\_index, close, array.get(a, 0))

EXAMPLE

//**@version=**6

indicator("array.new<color> example")

a = array.new<**color**>()

array.push(a, color.red)

array.push(a, color.green)

plot(close, color = array.get(a, close > open ? 1 : 0))

EXAMPLE

//**@version=**6

indicator("array.new<float> example")

length = 5

var a = array.new<**float**>(length, close)

if array.size(a) == length

array.remove(a, 0)

array.push(a, close)

plot(array.sum(a) / length, "SMA")

EXAMPLE

//**@version=**6

indicator("array.new<line> example")

// draw last 15 lines

var a = array.new<**line**>()

array.push(a, line.new(bar\_index - 1, close[1], bar\_index, close))

if array.size(a) > 15

ln = array.shift(a)

line.delete(ln)

RETURNS

The ID of an array object which may be used in other array.\*() functions.

REMARKS

An array index starts from 0.

If you want to initialize an array and specify all its elements at the same time, then use the function array.from.

SEE ALSO

[array.from](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from)[array.push](https://www.tradingview.com/pine-script-reference/v6/#fun_array.push)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.size](https://www.tradingview.com/pine-script-reference/v6/#fun_array.size)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.shift](https://www.tradingview.com/pine-script-reference/v6/#fun_array.shift)[array.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sum)

**array.percentile\_linear\_interpolation()**2 overloads

Returns the value for which the specified percentage of array values (percentile) are less than or equal to it, using linear interpolation.

SYNTAX & OVERLOADS

[array.percentile\_linear\_interpolation(id, percentage) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.percentile_linear_interpolation-0)

[array.percentile\_linear\_interpolation(id, percentage) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.percentile_linear_interpolation-1)

ARGUMENTS

**id (array<int/float>)** An array object.

**percentage (series int/float)** The percentage of values that must be equal or less than the returned value.

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.reverse](https://www.tradingview.com/pine-script-reference/v6/#fun_array.reverse)[order.ascending](https://www.tradingview.com/pine-script-reference/v6/#const_order.ascending)[order.descending](https://www.tradingview.com/pine-script-reference/v6/#const_order.descending)

**array.percentile\_nearest\_rank()**2 overloads

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

[array.percentile\_nearest\_rank(id, percentage) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.percentile_nearest_rank-0)

[array.percentile\_nearest\_rank(id, percentage) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.percentile_nearest_rank-1)

ARGUMENTS

**id (array<int/float>)** An array object.

**percentage (series int/float)** The percentage of values that must be equal or less than the returned value.

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.reverse](https://www.tradingview.com/pine-script-reference/v6/#fun_array.reverse)[order.ascending](https://www.tradingview.com/pine-script-reference/v6/#const_order.ascending)[order.descending](https://www.tradingview.com/pine-script-reference/v6/#const_order.descending)

**array.percentrank()**2 overloads

Returns the percentile rank of the element at the specified index.

SYNTAX & OVERLOADS

[array.percentrank(id, index) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.percentrank-0)

[array.percentrank(id, index) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.percentrank-1)

ARGUMENTS

**id (array<int/float>)** An array object.

**index (series int)** The index of the element for which the percentile rank should be calculated.

REMARKS

Percentile rank is the percentage of how many elements in the array are less than or equal to the reference value.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.reverse](https://www.tradingview.com/pine-script-reference/v6/#fun_array.reverse)[order.ascending](https://www.tradingview.com/pine-script-reference/v6/#const_order.ascending)[order.descending](https://www.tradingview.com/pine-script-reference/v6/#const_order.descending)

**array.pop()**

The function removes the last element from an array and returns its value.

SYNTAX

array.pop(id) → series <type>

ARGUMENTS

**id (any array type)** An array object.

EXAMPLE

//**@version=**6

indicator("array.pop example")

a = array.new\_float(5,high)

removedEl = array.pop(a)

plot(array.size(a))

plot(removedEl)

RETURNS

The value of the removed element.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.push](https://www.tradingview.com/pine-script-reference/v6/#fun_array.push)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.shift](https://www.tradingview.com/pine-script-reference/v6/#fun_array.shift)

**array.push()**

The function appends a value to an array.

SYNTAX

array.push(id, value) → void

ARGUMENTS

**id (any array type)** An array object.

**value (series <type of the array's elements>)** The value of the element added to the end of the array.

EXAMPLE

//**@version=**6

indicator("array.push example")

a = array.new\_float(5, 0)

array.push(a, open)

plot(array.get(a, 5))

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.pop](https://www.tradingview.com/pine-script-reference/v6/#fun_array.pop)[array.unshift](https://www.tradingview.com/pine-script-reference/v6/#fun_array.unshift)

**array.range()**2 overloads

The function returns the difference between the min and max values from a given array.

SYNTAX & OVERLOADS

[array.range(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.range-0)

[array.range(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.range-1)

ARGUMENTS

**id (array<int/float>)** An array object.

EXAMPLE

//**@version=**6

indicator("array.range example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

plot(array.range(a))

RETURNS

The difference between the min and max values in the array.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.min](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min)[array.max](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max)[array.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sum)

**array.remove()**

The function changes the contents of an array by removing the element with the specified index.

SYNTAX

array.remove(id, index) → series <type>

ARGUMENTS

**id (any array type)** An array object.

**index (series int)** The index of the element to remove.

EXAMPLE

//**@version=**6

indicator("array.remove example")

a = array.new\_float(5,high)

removedEl = array.remove(a, 0)

plot(array.size(a))

plot(removedEl)

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.push](https://www.tradingview.com/pine-script-reference/v6/#fun_array.push)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.pop](https://www.tradingview.com/pine-script-reference/v6/#fun_array.pop)[array.shift](https://www.tradingview.com/pine-script-reference/v6/#fun_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

array.reverse(id) → void

ARGUMENTS

**id (any array type)** An array object.

EXAMPLE

//**@version=**6

indicator("array.reverse example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

plot(array.get(a, 0))

array.reverse(a)

plot(array.get(a, 0))

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sort)[array.push](https://www.tradingview.com/pine-script-reference/v6/#fun_array.push)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_array.avg)

**array.set()**

The function sets the value of the element at the specified index.

SYNTAX

array.set(id, index, value) → void

ARGUMENTS

**id (any array type)** An array object.

**index (series int)** The index of the element to be modified.

**value (series <type of the array's elements>)** The new value to be set.

EXAMPLE

//**@version=**6

indicator("array.set example")

a = array.new\_float(10)

for i = 0 to 9

array.set(a, i, close[i])

plot(array.sum(a) / 10)

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)

**array.shift()**

The function removes an array's first element and returns its value.

SYNTAX

array.shift(id) → series <type>

ARGUMENTS

**id (any array type)** An array object.

EXAMPLE

//**@version=**6

indicator("array.shift example")

a = array.new\_float(5,high)

removedEl = array.shift(a)

plot(array.size(a))

plot(removedEl)

RETURNS

The value of the removed element.

SEE ALSO

[array.unshift](https://www.tradingview.com/pine-script-reference/v6/#fun_array.unshift)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.push](https://www.tradingview.com/pine-script-reference/v6/#fun_array.push)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.includes](https://www.tradingview.com/pine-script-reference/v6/#fun_array.includes)

**array.size()**

The function returns the number of elements in an array.

SYNTAX

array.size(id) → series int

ARGUMENTS

**id (any array type)** An array object.

EXAMPLE

//**@version=**6

indicator("array.size example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

// note that changes in slice also modify original array

slice = array.slice(a, 0, 5)

array.push(slice, open)

// size was changed in slice and in original array

plot(array.size(a))

plot(array.size(slice))

RETURNS

The number of elements in the array.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sum)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_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

array.slice(id, index\_from, index\_to) → array<type>

ARGUMENTS

**id (any array type)** An array object.

**index\_from (series int)** Zero-based index at which to begin extraction.

**index\_to (series int)** Zero-based index before which to end extraction. The function extracts up to but not including the element with this index.

EXAMPLE

//**@version=**6

indicator("array.slice example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

// take elements from 0 to 4

// \*note that changes in slice also modify original array

slice = array.slice(a, 0, 5)

plot(array.sum(a) / 10)

plot(array.sum(slice) / 5)

RETURNS

A shallow copy of an array's slice.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sort)

**array.some()**

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if at least one element of the id array is [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SYNTAX

array.some(id) → series bool

ARGUMENTS

**id (array<bool>)** An array object.

REMARKS

This function also works with arrays of [int](https://www.tradingview.com/pine-script-reference/v6/#type_int) and [float](https://www.tradingview.com/pine-script-reference/v6/#type_float) types, in which case zero values are considered [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), and all others [true](https://www.tradingview.com/pine-script-reference/v6/#const_true).

SEE ALSO

[array.every](https://www.tradingview.com/pine-script-reference/v6/#fun_array.every)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)

**array.sort()**

The function sorts the elements of an array.

SYNTAX

array.sort(id, order) → void

ARGUMENTS

**id (array<int/float/string>)** An array object.

**order (series sort\_order)** The sort order: order.ascending (default) or order.descending.

EXAMPLE

//**@version=**6

indicator("array.sort example")

a = array.new\_float(0,0)

for i = 0 to 5

array.push(a, high[i])

array.sort(a, order.descending)

if barstate.islast

label.new(bar\_index, close, str.tostring(a))

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.reverse](https://www.tradingview.com/pine-script-reference/v6/#fun_array.reverse)[order.ascending](https://www.tradingview.com/pine-script-reference/v6/#const_order.ascending)[order.descending](https://www.tradingview.com/pine-script-reference/v6/#const_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

array.sort\_indices(id, order) → array<int>

ARGUMENTS

**id (array<int/float/string>)** An array object.

**order (series sort\_order)** The sort order: order.ascending or order.descending. Optional. The default is order.ascending.

EXAMPLE

//**@version=**6

indicator("array.sort\_indices")

a = array.from(5, -2, 0, 9, 1)

sortedIndices = array.sort\_indices(a) // [1, 2, 4, 0, 3]

indexOfSmallestValue = array.get(sortedIndices, 0) // 1

smallestValue = array.get(a, indexOfSmallestValue) // -2

plot(smallestValue)

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.slice](https://www.tradingview.com/pine-script-reference/v6/#fun_array.slice)[array.reverse](https://www.tradingview.com/pine-script-reference/v6/#fun_array.reverse)[order.ascending](https://www.tradingview.com/pine-script-reference/v6/#const_order.ascending)[order.descending](https://www.tradingview.com/pine-script-reference/v6/#const_order.descending)

**array.standardize()**2 overloads

The function returns the array of standardized elements.

SYNTAX & OVERLOADS

[array.standardize(id) → array<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.standardize-0)

[array.standardize(id) → array<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.standardize-1)

ARGUMENTS

**id (array<int/float>)** An array object.

EXAMPLE

//**@version=**6

indicator("array.standardize example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

b = array.standardize(a)

plot(array.min(b))

plot(array.max(b))

RETURNS

The array of standardized elements.

SEE ALSO

[array.max](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max)[array.min](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min)[array.mode](https://www.tradingview.com/pine-script-reference/v6/#fun_array.mode)[array.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_array.avg)[array.variance](https://www.tradingview.com/pine-script-reference/v6/#fun_array.variance)[array.stdev](https://www.tradingview.com/pine-script-reference/v6/#fun_array.stdev)

**array.stdev()**2 overloads

The function returns the standard deviation of an array's elements.

SYNTAX & OVERLOADS

[array.stdev(id, biased) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.stdev-0)

[array.stdev(id, biased) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.stdev-1)

ARGUMENTS

**id (array<int/float>)** An array object.

**biased (series bool)** Determines which estimate should be used. Optional. The default is true.

EXAMPLE

//**@version=**6

indicator("array.stdev example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

plot(array.stdev(a))

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.max](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max)[array.min](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min)[array.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_array.avg)

**array.sum()**2 overloads

The function returns the sum of an array's elements.

SYNTAX & OVERLOADS

[array.sum(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sum-0)

[array.sum(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sum-1)

ARGUMENTS

**id (array<int/float>)** An array object.

EXAMPLE

//**@version=**6

indicator("array.sum example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

plot(array.sum(a))

RETURNS

The sum of the array's elements.

SEE ALSO

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.max](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max)[array.min](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min)

**array.unshift()**

The function inserts the value at the beginning of the array.

SYNTAX

array.unshift(id, value) → void

ARGUMENTS

**id (any array type)** An array object.

**value (series <type of the array's elements>)** The value to add to the start of the array.

EXAMPLE

//**@version=**6

indicator("array.unshift example")

a = array.new\_float(5, 0)

array.unshift(a, open)

plot(array.get(a, 0))

SEE ALSO

[array.shift](https://www.tradingview.com/pine-script-reference/v6/#fun_array.shift)[array.set](https://www.tradingview.com/pine-script-reference/v6/#fun_array.set)[array.insert](https://www.tradingview.com/pine-script-reference/v6/#fun_array.insert)[array.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_array.remove)[array.indexof](https://www.tradingview.com/pine-script-reference/v6/#fun_array.indexof)

**array.variance()**2 overloads

The function returns the variance of an array's elements.

SYNTAX & OVERLOADS

[array.variance(id, biased) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.variance-0)

[array.variance(id, biased) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_array.variance-1)

ARGUMENTS

**id (array<int/float>)** An array object.

**biased (series bool)** Determines which estimate should be used. Optional. The default is true.

EXAMPLE

//**@version=**6

indicator("array.variance example")

a = array.new\_float(0)

for i = 0 to 9

array.push(a, close[i])

plot(array.variance(a))

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

[array.new\_float](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new_float)[array.stdev](https://www.tradingview.com/pine-script-reference/v6/#fun_array.stdev)[array.min](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min)[array.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_array.avg)[array.covariance](https://www.tradingview.com/pine-script-reference/v6/#fun_array.covariance)

**barcolor()**

Set color of bars.

SYNTAX

barcolor(color, offset, editable, show\_last, title, display) → void

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.

**offset (simple int)** Shifts the color series to the left or to the right on the given number of bars. Default is 0.

**editable (const bool)** If true then barcolor style will be editable in Format dialog. Default is true.

**show\_last (input int)** Optional. The number of bars, counting backwards from the most recent bar, on which the function can draw.

**title (const string)** Title of the barcolor. Optional argument.

**display (input plot\_simple\_display)** Controls where the barcolor is displayed. Possible values are: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

EXAMPLE

//**@version=**6

indicator("barcolor example", overlay=true)

barcolor(close < open ? color.black : color.white)

SEE ALSO

[bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_bgcolor)[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[fill](https://www.tradingview.com/pine-script-reference/v6/#fun_fill)

**bgcolor()**

Fill background of bars with specified color.

SYNTAX

bgcolor(color, offset, editable, show\_last, title, display, force\_overlay) → void

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.

**offset (simple int)** Shifts the color series to the left or to the right on the given number of bars. Default is 0.

**editable (const bool)** If true then bgcolor style will be editable in Format dialog. Default is true.

**show\_last (input int)** Optional. The number of bars, counting backwards from the most recent bar, on which the function can draw.

**title (const string)** Title of the bgcolor. Optional argument.

**display (input plot\_simple\_display)** Controls where the bgcolor is displayed. Possible values are: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the plotted results will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("bgcolor example", overlay=true)

bgcolor(close < open ? color.new(color.red,70) : color.new(color.green, 70))

SEE ALSO

[barcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_barcolor)[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[fill](https://www.tradingview.com/pine-script-reference/v6/#fun_fill)

**bool()**4 overloads

Converts the x value to a [bool](https://www.tradingview.com/pine-script-reference/v6/#type_bool) value. Returns [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) if x is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), or an [int](https://www.tradingview.com/pine-script-reference/v6/#type_int)/[float](https://www.tradingview.com/pine-script-reference/v6/#type_float) value equal to 0. Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) for all other possible values.

SYNTAX & OVERLOADS

[bool(x) → const bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool-0)

[bool(x) → input bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool-1)

[bool(x) → simple bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool-2)

[bool(x) → series bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool-3)

ARGUMENTS

**x (simple int/float/bool)** The value to convert to the specified type, usually [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

RETURNS

The value of the argument after casting to bool.

SEE ALSO

[float](https://www.tradingview.com/pine-script-reference/v6/#fun_float)[int](https://www.tradingview.com/pine-script-reference/v6/#fun_int)[color](https://www.tradingview.com/pine-script-reference/v6/#fun_color)[string](https://www.tradingview.com/pine-script-reference/v6/#fun_string)[line](https://www.tradingview.com/pine-script-reference/v6/#fun_line)[label](https://www.tradingview.com/pine-script-reference/v6/#fun_label)

**box()**

Casts na to box.

SYNTAX

box(x) → series box

ARGUMENTS

**x (series box)** The value to convert to the specified type, usually [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

RETURNS

The value of the argument after casting to box.

SEE ALSO

[float](https://www.tradingview.com/pine-script-reference/v6/#fun_float)[int](https://www.tradingview.com/pine-script-reference/v6/#fun_int)[bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool)[color](https://www.tradingview.com/pine-script-reference/v6/#fun_color)[string](https://www.tradingview.com/pine-script-reference/v6/#fun_string)[line](https://www.tradingview.com/pine-script-reference/v6/#fun_line)[label](https://www.tradingview.com/pine-script-reference/v6/#fun_label)

**box.copy()**

Clones the box object.

SYNTAX

box.copy(id) → series box

ARGUMENTS

**id (series box)** Box object.

EXAMPLE

//**@version=**6

indicator('Last 50 bars price ranges', overlay = true)

LOOKBACK = 50

highest = ta.highest(LOOKBACK)

lowest = ta.lowest(LOOKBACK)

if barstate.islastconfirmedhistory

var BoxLast = box.new(bar\_index[LOOKBACK], highest, bar\_index, lowest, bgcolor = color.new(color.green, 80))

var BoxPrev = box.copy(BoxLast)

box.set\_lefttop(BoxPrev, bar\_index[LOOKBACK \* 2], highest[50])

box.set\_rightbottom(BoxPrev, bar\_index[LOOKBACK], lowest[50])

box.set\_bgcolor(BoxPrev, color.new(color.red, 80))

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_box.delete)

**box.delete()**

Deletes the specified box object. If it has already been deleted, does nothing.

SYNTAX

box.delete(id) → void

ARGUMENTS

**id (series box)** A box object to delete.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)

**box.get\_bottom()**

Returns the price value of the bottom border of the box.

SYNTAX

box.get\_bottom(id) → series float

ARGUMENTS

**id (series box)** A box object.

RETURNS

The price value.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.set\_bottom](https://www.tradingview.com/pine-script-reference/v6/#fun_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

box.get\_left(id) → series int

ARGUMENTS

**id (series box)** A box object.

RETURNS

A bar index or a UNIX timestamp (in milliseconds).

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.set\_left](https://www.tradingview.com/pine-script-reference/v6/#fun_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

box.get\_right(id) → series int

ARGUMENTS

**id (series box)** A box object.

RETURNS

A bar index or a UNIX timestamp (in milliseconds).

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.set\_right](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_right)

**box.get\_top()**

Returns the price value of the top border of the box.

SYNTAX

box.get\_top(id) → series float

ARGUMENTS

**id (series box)** A box object.

RETURNS

The price value.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.set\_top](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_top)

**box.new()**2 overloads

Creates a new box object.

SYNTAX & OVERLOADS

[box.new(top\_left, bottom\_right, border\_color, border\_width, border\_style, extend, xloc, bgcolor, text, text\_size, text\_color, text\_halign, text\_valign, text\_wrap, text\_font\_family, force\_overlay, text\_formatting) → series box](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new-0)

[box.new(left, top, right, bottom, border\_color, border\_width, border\_style, extend, xloc, bgcolor, text, text\_size, text\_color, text\_halign, text\_valign, text\_wrap, text\_font\_family, force\_overlay, text\_formatting) → series box](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new-1)

ARGUMENTS

**top\_left (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object that specifies the top-left corner location of the box.

**bottom\_right (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object that specifies the bottom-right corner location of the box.

**border\_color (series color)** Color of the four borders. Optional. The default is [color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue).

**border\_width (series int)** Width of the four borders, in pixels. Optional. The default is 1 pixel.

**border\_style (series string)** Style of the four borders. Possible values: [line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid), [line.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dotted), [line.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dashed). Optional. The default value is [line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid).

**extend (series string)** When [extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none) is used, the horizontal borders start at the left border and end at the right border. With [extend.left](https://www.tradingview.com/pine-script-reference/v6/#const_extend.left) or [extend.right](https://www.tradingview.com/pine-script-reference/v6/#const_extend.right), the horizontal borders are extended indefinitely to the left or right of the box, respectively. With [extend.both](https://www.tradingview.com/pine-script-reference/v6/#const_extend.both), the horizontal borders are extended on both sides. Optional. The default value is [extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none).

**xloc (series string)** Determines whether the arguments to 'left' and 'right' are a bar index or a time value. If xloc = [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index), the arguments must be a bar index. If xloc = [xloc.bar\_time](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_time), the arguments must be a UNIX time. Possible values: [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index) and [xloc.bar\_time](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_time). Optional. The default is [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index).

**bgcolor (series color)** Background color of the box. Optional. The default is [color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue).

**text (series string)** The text to be displayed inside the box. Optional. The default is empty string.

**text\_size (series int/string)** Optional. Size of the box's text. The size can be any positive integer, or one of the size.\* built-in constant strings. The constant strings and their equivalent integer values are: [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto) (0), [size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny) (8), [size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small) (10), [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal) (14), [size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large) (20), [size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge) (36). The default value is [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto) or 0.

**text\_color (series color)** The color of the text. Optional. The default is [color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black).

**text\_halign (series string)** The horizontal alignment of the box's text. Optional. The default value is [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center). Possible values: [text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left), [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center), [text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_right).

**text\_valign (series string)** The vertical alignment of the box's text. Optional. The default value is [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center). Possible values: [text.align\_top](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_top), [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center), [text.align\_bottom](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_bottom).

**text\_wrap (series string)** Optional. Whether to wrap text. Wrapped text starts a new line when it reaches the side of the box. Wrapped text lower than the bottom of the box is not displayed. Unwrapped text stays on a single line and *is displayed* past the width of the box if it is too long. If the text\_size is 0 or [text.wrap\_auto](https://www.tradingview.com/pine-script-reference/v6/#const_text.wrap_auto), this setting has no effect. The default value is [text.wrap\_none](https://www.tradingview.com/pine-script-reference/v6/#const_text.wrap_none). Possible values: [text.wrap\_none](https://www.tradingview.com/pine-script-reference/v6/#const_text.wrap_none), [text.wrap\_auto](https://www.tradingview.com/pine-script-reference/v6/#const_text.wrap_auto).

**text\_font\_family (series string)** The font family of the text. Optional. The default value is [font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default). Possible values: [font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default), [font.family\_monospace](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_monospace).

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the drawing will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**text\_formatting (const text\_format)** The formatting of the displayed text. Formatting options support addition. For example, text.format\_bold + text.format\_italic will make the text both bold and italicized. Possible values: [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none), [text.format\_bold](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_bold), [text.format\_italic](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_italic). Optional. The default is [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none).

EXAMPLE

//**@version=**6

indicator("box.new")

var b = box.new(time, open, time + 60 \* 60 \* 24, close, xloc=xloc.bar\_time, border\_style=line.style\_dashed)

box.set\_lefttop(b, time, 100)

box.set\_rightbottom(b, time + 60 \* 60 \* 24, 500)

box.set\_bgcolor(b, color.green)

RETURNS

The ID of a box object which may be used in box.set\_\*() and box.get\_\*() functions.

SEE ALSO

[box.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_box.delete)[box.get\_left](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_left)[box.get\_top](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_top)[box.get\_right](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_right)[box.get\_bottom](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_bottom)[box.set\_top\_left\_point](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_top_left_point)[box.set\_left](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_left)[box.set\_top](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_top)[box.set\_bottom\_right\_point](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_bottom_right_point)[box.set\_right](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_right)[box.set\_bottom](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_bottom)[box.set\_border\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_border_color)[box.set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_bgcolor)[box.set\_border\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_border_width)[box.set\_border\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_border_style)[box.set\_extend](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_extend)[box.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text)[box.set\_text\_formatting](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_formatting)

**box.set\_bgcolor()**

Sets the background color of the box.

SYNTAX

box.set\_bgcolor(id, color) → void

ARGUMENTS

**id (series box)** A box object.

**color (series color)** New background color.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)

**box.set\_border\_color()**

Sets the border color of the box.

SYNTAX

box.set\_border\_color(id, color) → void

ARGUMENTS

**id (series box)** A box object.

**color (series color)** New border color.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)

**box.set\_border\_style()**

Sets the border style of the box.

SYNTAX

box.set\_border\_style(id, style) → void

ARGUMENTS

**id (series box)** A box object.

**style (series string)** New border style.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid)

**box.set\_border\_width()**

Sets the border width of the box.

SYNTAX

box.set\_border\_width(id, width) → void

ARGUMENTS

**id (series box)** A box object.

**width (series int)** Width of the four borders, in pixels.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)

**box.set\_bottom()**

Sets the bottom coordinate of the box.

SYNTAX

box.set\_bottom(id, bottom) → void

ARGUMENTS

**id (series box)** A box object.

**bottom (series int/float)** Price value of the bottom border.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.get\_bottom](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_bottom)

**box.set\_bottom\_right\_point()**

Sets the bottom-right corner location of the id box to point.

SYNTAX

box.set\_bottom\_right\_point(id, point) → void

ARGUMENTS

**id (series box)** A [box](https://www.tradingview.com/pine-script-reference/v6/#type_box) object.

**point (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object.

**box.set\_extend()**

Sets extending type of the border of this box object. When [extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none) is used, the horizontal borders start at the left border and end at the right border. With [extend.left](https://www.tradingview.com/pine-script-reference/v6/#const_extend.left) or [extend.right](https://www.tradingview.com/pine-script-reference/v6/#const_extend.right), the horizontal borders are extended indefinitely to the left or right of the box, respectively. With [extend.both](https://www.tradingview.com/pine-script-reference/v6/#const_extend.both), the horizontal borders are extended on both sides.

SYNTAX

box.set\_extend(id, extend) → void

ARGUMENTS

**id (series box)** A box object.

**extend (series string)** New extending type.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none)

**box.set\_left()**

Sets the left coordinate of the box.

SYNTAX

box.set\_left(id, left) → void

ARGUMENTS

**id (series box)** A box object.

**left (series int)** Bar index or bar time of the left border. Note that objects positioned using [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index) cannot be drawn further than 500 bars into the future.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.get\_left](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_left)

**box.set\_lefttop()**

Sets the left and top coordinates of the box.

SYNTAX

box.set\_lefttop(id, left, top) → void

ARGUMENTS

**id (series box)** A box object.

**left (series int)** Bar index or bar time of the left border.

**top (series int/float)** Price value of the top border.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.get\_left](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_left)[box.get\_top](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_top)

**box.set\_right()**

Sets the right coordinate of the box.

SYNTAX

box.set\_right(id, right) → void

ARGUMENTS

**id (series box)** A box object.

**right (series int)** Bar index or bar time of the right border. Note that objects positioned using [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index) cannot be drawn further than 500 bars into the future.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.get\_right](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_right)

**box.set\_rightbottom()**

Sets the right and bottom coordinates of the box.

SYNTAX

box.set\_rightbottom(id, right, bottom) → void

ARGUMENTS

**id (series box)** A box object.

**right (series int)** Bar index or bar time of the right border.

**bottom (series int/float)** Price value of the bottom border.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.get\_right](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_right)[box.get\_bottom](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_bottom)

**box.set\_text()**

The function sets the text in the box.

SYNTAX

box.set\_text(id, text) → void

ARGUMENTS

**id (series box)** A box object.

**text (series string)** The text to be displayed inside the box.

SEE ALSO

[box.set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_color)[box.set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_size)[box.set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_valign)[box.set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_halign)[box.set\_text\_formatting](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_formatting)

**box.set\_text\_color()**

The function sets the color of the text inside the box.

SYNTAX

box.set\_text\_color(id, text\_color) → void

ARGUMENTS

**id (series box)** A box object.

**text\_color (series color)** The color of the text.

SEE ALSO

[box.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text)[box.set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_size)[box.set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_valign)[box.set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_halign)

**box.set\_text\_font\_family()**

The function sets the font family of the text inside the box.

SYNTAX

box.set\_text\_font\_family(id, text\_font\_family) → void

ARGUMENTS

**id (series box)** A box object.

**text\_font\_family (series string)** The font family of the text. Possible values: [font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default), [font.family\_monospace](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_monospace).

EXAMPLE

//**@version=**6

indicator("Example of setting the box font")

if barstate.islastconfirmedhistory

b = box.new(bar\_index, open-ta.tr, bar\_index-50, open-ta.tr\*5, text="monospace")

box.set\_text\_font\_family(b, font.family\_monospace)

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default)[font.family\_monospace](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_monospace)

**box.set\_text\_formatting()**

Sets the formatting attributes the drawing applies to displayed text.

SYNTAX

box.set\_text\_formatting(id, text\_formatting) → void

ARGUMENTS

**id (series box)** A box object.

**text\_formatting (const text\_format)** The formatting of the displayed text. Formatting options support addition. For example, text.format\_bold + text.format\_italic will make the text both bold and italicized. Possible values: [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none), [text.format\_bold](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_bold), [text.format\_italic](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_italic).

SEE ALSO

[box.set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_color)[box.set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_size)[box.set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_valign)[box.set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_halign)[box.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text)

**box.set\_text\_halign()**

The function sets the horizontal alignment of the box's text.

SYNTAX

box.set\_text\_halign(id, text\_halign) → void

ARGUMENTS

**id (series box)** A box object.

**text\_halign (series string)** The horizontal alignment of a box's text. Possible values: [text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left), [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center), [text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_right).

SEE ALSO

[box.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text)[box.set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_size)[box.set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_valign)[box.set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_color)

**box.set\_text\_size()**

The function sets the size of the box's text.

SYNTAX

box.set\_text\_size(id, text\_size) → void

ARGUMENTS

**id (series box)** A box object.

**text\_size (series int/string)** Size of the box's text. The size can be any positive integer, or one of the size.\* built-in constant strings. The constant strings and their equivalent integer values are: [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto) (0), [size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny) (8), [size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small) (10), [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal) (14), [size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large) (20), [size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge) (36).

SEE ALSO

[box.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text)[box.set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_color)[box.set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_valign)[box.set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_halign)

**box.set\_text\_valign()**

The function sets the vertical alignment of a box's text.

SYNTAX

box.set\_text\_valign(id, text\_valign) → void

ARGUMENTS

**id (series box)** A box object.

**text\_valign (series string)** The vertical alignment of the box's text. Possible values: [text.align\_top](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_top), [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center), [text.align\_bottom](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_bottom).

SEE ALSO

[box.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text)[box.set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_size)[box.set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_color)[box.set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_halign)

**box.set\_text\_wrap()**

The function sets the mode of wrapping of the text inside the box.

SYNTAX

box.set\_text\_wrap(id, text\_wrap) → void

ARGUMENTS

**id (series box)** A box object.

**text\_wrap (series string)** Whether to wrap text. Wrapped text starts a new line when it reaches the side of the box. Wrapped text lower than the bottom of the box is not displayed. Unwrapped text stays on a single line and *is displayed* past the width of the box if it is too long. If the text\_size is 0 or [text.wrap\_auto](https://www.tradingview.com/pine-script-reference/v6/#const_text.wrap_auto), this setting has no effect. Possible values: [text.wrap\_none](https://www.tradingview.com/pine-script-reference/v6/#const_text.wrap_none), [text.wrap\_auto](https://www.tradingview.com/pine-script-reference/v6/#const_text.wrap_auto).

SEE ALSO

[box.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text)[box.set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_size)[box.set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_valign)[box.set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_halign)[box.set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_box.set_text_color)

**box.set\_top()**

Sets the top coordinate of the box.

SYNTAX

box.set\_top(id, top) → void

ARGUMENTS

**id (series box)** A box object.

**top (series int/float)** Price value of the top border.

SEE ALSO

[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new)[box.get\_top](https://www.tradingview.com/pine-script-reference/v6/#fun_box.get_top)

**box.set\_top\_left\_point()**

Sets the top-left corner location of the id box to point.

SYNTAX

box.set\_top\_left\_point(id, point) → void

ARGUMENTS

**id (series box)** A [box](https://www.tradingview.com/pine-script-reference/v6/#type_box) object.

**point (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object.

**chart.point.copy()**

Creates a copy of a [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object with the specified id.

SYNTAX

chart.point.copy(id) → chart.point

ARGUMENTS

**id (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object.

**chart.point.from\_index()**

Returns a [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object with index as its x-coordinate and price as its y-coordinate.

SYNTAX

chart.point.from\_index(index, price) → chart.point

ARGUMENTS

**index (series int)** The x-coordinate of the point, expressed as a bar index value.

**price (series int/float)** The y-coordinate of the point.

REMARKS

The time field values of [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) instances returned from this function will be [na](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object with time as its x-coordinate and price as its y-coordinate.

SYNTAX

chart.point.from\_time(time, price) → chart.point

ARGUMENTS

**time (series int)** The x-coordinate of the point, expressed as a UNIX time value, in milliseconds.

**price (series int/float)** The y-coordinate of the point.

REMARKS

The index field values of [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) instances returned from this function will be [na](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object with the specified time, index, and price.

SYNTAX

chart.point.new(time, index, price) → chart.point

ARGUMENTS

**time (series int)** The x-coordinate of the point, expressed as a UNIX time value, in milliseconds.

**index (series int)** The x-coordinate of the point, expressed as a bar index value.

**price (series int/float)** The y-coordinate of the point.

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.

It's important to note that this function does not verify that the time and index values refer to the same bar.

SEE ALSO

[polyline.new](https://www.tradingview.com/pine-script-reference/v6/#fun_polyline.new)

**chart.point.now()**

Returns a [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object with price as the y-coordinate

SYNTAX

chart.point.now(price) → chart.point

ARGUMENTS

**price (series int/float)** The y-coordinate of the point. Optional. The default is [close](https://www.tradingview.com/pine-script-reference/v6/#var_close).

REMARKS

The [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_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()**4 overloads

Casts na to color

SYNTAX & OVERLOADS

[color(x) → const color](https://www.tradingview.com/pine-script-reference/v6/#fun_color-0)

[color(x) → input color](https://www.tradingview.com/pine-script-reference/v6/#fun_color-1)

[color(x) → simple color](https://www.tradingview.com/pine-script-reference/v6/#fun_color-2)

[color(x) → series color](https://www.tradingview.com/pine-script-reference/v6/#fun_color-3)

ARGUMENTS

**x (const color)** The value to convert to the specified type, usually [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

RETURNS

The value of the argument after casting to color.

SEE ALSO

[float](https://www.tradingview.com/pine-script-reference/v6/#fun_float)[int](https://www.tradingview.com/pine-script-reference/v6/#fun_int)[bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool)[string](https://www.tradingview.com/pine-script-reference/v6/#fun_string)[line](https://www.tradingview.com/pine-script-reference/v6/#fun_line)[label](https://www.tradingview.com/pine-script-reference/v6/#fun_label)

**color.b()**4 overloads

Retrieves the value of the color's blue component.

SYNTAX & OVERLOADS

[color.b(color) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.b-0)

[color.b(color) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.b-1)

[color.b(color) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.b-2)

[color.b(color) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.b-3)

ARGUMENTS

**color (const color)** Color.

EXAMPLE

//**@version=**6

indicator("color.b", overlay=true)

plot(color.b(color.blue))

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

color.from\_gradient(value, bottom\_value, top\_value, bottom\_color, top\_color) → series color

ARGUMENTS

**value (series int/float)** Value to calculate the position-dependent color.

**bottom\_value (series int/float)** Bottom position value corresponding to bottom\_color.

**top\_value (series int/float)** Top position value corresponding to top\_color.

**bottom\_color (series color)** Bottom position color.

**top\_color (series color)** Top position color.

EXAMPLE

//**@version=**6

indicator("color.from\_gradient", overlay=true)

color1 = color.from\_gradient(close, low, high, color.yellow, color.lime)

color2 = color.from\_gradient(ta.rsi(close, 7), 0, 100, color.rgb(255, 0, 0), color.rgb(0, 255, 0, 50))

plot(close, color=color1)

plot(ta.rsi(close,7), color=color2)

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](https://www.tradingview.com/pine-script-docs/concepts/colors/#color-selection-through-script-settings) for more information.

**color.g()**4 overloads

Retrieves the value of the color's green component.

SYNTAX & OVERLOADS

[color.g(color) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.g-0)

[color.g(color) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.g-1)

[color.g(color) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.g-2)

[color.g(color) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.g-3)

ARGUMENTS

**color (const color)** Color.

EXAMPLE

//**@version=**6

indicator("color.g", overlay=true)

plot(color.g(color.green))

RETURNS

The value (0 to 255) of the color's green component.

**color.new()**4 overloads

Function color applies the specified transparency to the given color.

SYNTAX & OVERLOADS

[color.new(color, transp) → const color](https://www.tradingview.com/pine-script-reference/v6/#fun_color.new-0)

[color.new(color, transp) → input color](https://www.tradingview.com/pine-script-reference/v6/#fun_color.new-1)

[color.new(color, transp) → simple color](https://www.tradingview.com/pine-script-reference/v6/#fun_color.new-2)

[color.new(color, transp) → series color](https://www.tradingview.com/pine-script-reference/v6/#fun_color.new-3)

ARGUMENTS

**color (const color)** Color to apply transparency to.

**transp (const int/float)** Possible values are from 0 (not transparent) to 100 (invisible).

EXAMPLE

//**@version=**6

indicator("color.new", overlay=true)

plot(close, color=color.new(color.red, 50))

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](https://www.tradingview.com/pine-script-docs/concepts/colors/#color-selection-through-script-settings) for more information.

**color.r()**4 overloads

Retrieves the value of the color's red component.

SYNTAX & OVERLOADS

[color.r(color) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.r-0)

[color.r(color) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.r-1)

[color.r(color) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.r-2)

[color.r(color) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.r-3)

ARGUMENTS

**color (const color)** Color.

EXAMPLE

//**@version=**6

indicator("color.r", overlay=true)

plot(color.r(color.red))

RETURNS

The value (0 to 255) of the color's red component.

**color.rgb()**4 overloads

Creates a new color with transparency using the RGB color model.

SYNTAX & OVERLOADS

[color.rgb(red, green, blue, transp) → const color](https://www.tradingview.com/pine-script-reference/v6/#fun_color.rgb-0)

[color.rgb(red, green, blue, transp) → input color](https://www.tradingview.com/pine-script-reference/v6/#fun_color.rgb-1)

[color.rgb(red, green, blue, transp) → simple color](https://www.tradingview.com/pine-script-reference/v6/#fun_color.rgb-2)

[color.rgb(red, green, blue, transp) → series color](https://www.tradingview.com/pine-script-reference/v6/#fun_color.rgb-3)

ARGUMENTS

**red (const int/float)** Red color component. Possible values are from 0 to 255.

**green (const int/float)** Green color component. Possible values are from 0 to 255.

**blue (const int/float)** Blue color component. Possible values are from 0 to 255.

**transp (const int/float)** Optional. Color transparency. Possible values are from 0 (opaque) to 100 (invisible). Default value is 0.

EXAMPLE

//**@version=**6

indicator("color.rgb", overlay=true)

plot(close, color=color.rgb(255, 0, 0, 50))

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](https://www.tradingview.com/pine-script-docs/concepts/colors/#color-selection-through-script-settings) for more information.

**color.t()**4 overloads

Retrieves the color's transparency.

SYNTAX & OVERLOADS

[color.t(color) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.t-0)

[color.t(color) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.t-1)

[color.t(color) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.t-2)

[color.t(color) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_color.t-3)

ARGUMENTS

**color (const color)** Color.

EXAMPLE

//**@version=**6

indicator("color.t", overlay=true)

plot(color.t(color.new(color.red, 50)))

RETURNS

The value (0-100) of the color's transparency.

**dayofmonth()**2 overloads

SYNTAX & OVERLOADS

[dayofmonth(time) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofmonth-0)

[dayofmonth(time, timezone) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofmonth-1)

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.

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 UTC-4) this value can be lower by 1 than the day of the trading day.

SEE ALSO

[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#var_dayofmonth)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[year](https://www.tradingview.com/pine-script-reference/v6/#fun_year)[month](https://www.tradingview.com/pine-script-reference/v6/#fun_month)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#fun_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#fun_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#fun_second)

**dayofweek()**2 overloads

SYNTAX & OVERLOADS

[dayofweek(time) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek-0)

[dayofweek(time, timezone) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek-1)

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.

UNIX time is the number of milliseconds that have elapsed since 00:00:00 UTC, 1 January 1970.

SEE ALSO

[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#var_dayofweek)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[year](https://www.tradingview.com/pine-script-reference/v6/#fun_year)[month](https://www.tradingview.com/pine-script-reference/v6/#fun_month)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofmonth)[hour](https://www.tradingview.com/pine-script-reference/v6/#fun_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#fun_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#fun_second)

**fill()**3 overloads

Fills background between two plots or hlines with a given color.

SYNTAX & OVERLOADS

[fill(hline1, hline2, color, title, editable, fillgaps, display) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_fill-0)

[fill(plot1, plot2, color, title, editable, show\_last, fillgaps, display) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_fill-1)

[fill(plot1, plot2, top\_value, bottom\_value, top\_color, bottom\_color, title, display, fillgaps, editable) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_fill-2)

ARGUMENTS

**hline1 (hline)** The first hline object. Required argument.

**hline2 (hline)** The second hline object. Required argument.

**color (series color)** Color of the background fill. You can use constants like 'color=color.red' or 'color=#ff001a' as well as complex expressions like 'color = close >= open ? color.green : color.red'. Optional argument.

**title (const string)** Title of the created fill object. Optional argument.

**editable (const bool)** If true then fill style will be editable in Format dialog. Default is true.

**fillgaps (const bool)** Controls continuing fills on gaps, i.e., when one of the plot() calls returns an na value. When true, the last fill will continue on gaps. The default is false.

**display (input plot\_simple\_display)** Controls where the fill is displayed. Possible values are: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

Fill between two horizontal lines

EXAMPLE

//**@version=**6

indicator("Fill between hlines", overlay = false)

h1 = hline(20)

h2 = hline(10)

fill(h1, h2, color = color.new(color.blue, 90))

Fill between two plots

EXAMPLE

//**@version=**6

indicator("Fill between plots", overlay = true)

p1 = plot(open)

p2 = plot(close)

fill(p1, p2, color = color.new(color.green, 90))

Gradient fill between two horizontal lines

EXAMPLE

//**@version=**6

indicator("Gradient Fill between hlines", overlay = false)

topVal = input.int(100)

botVal = input.int(0)

topCol = input.color(color.red)

botCol = input.color(color.blue)

topLine = hline(100, color = topCol, linestyle = hline.style\_solid)

botLine = hline(0, color = botCol, linestyle = hline.style\_solid)

fill(topLine, botLine, topVal, botVal, topCol, botCol)

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[barcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_barcolor)[bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_bgcolor)[hline](https://www.tradingview.com/pine-script-reference/v6/#fun_hline)[color.new](https://www.tradingview.com/pine-script-reference/v6/#fun_color.new)

**fixnan()**3 overloads

For a given series replaces NaN values with previous nearest non-NaN value.

SYNTAX & OVERLOADS

[fixnan(source) → series color](https://www.tradingview.com/pine-script-reference/v6/#fun_fixnan-0)

[fixnan(source) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_fixnan-1)

[fixnan(source) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_fixnan-2)

ARGUMENTS

**source (series color)** Source used for the calculation.

RETURNS

Series without na gaps.

SEE ALSO

[na](https://www.tradingview.com/pine-script-reference/v6/#fun_na)[na](https://www.tradingview.com/pine-script-reference/v6/#var_na)[nz](https://www.tradingview.com/pine-script-reference/v6/#fun_nz)

**float()**4 overloads

Casts na to float

SYNTAX & OVERLOADS

[float(x) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_float-0)

[float(x) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_float-1)

[float(x) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_float-2)

[float(x) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_float-3)

ARGUMENTS

**x (const int/float)** The value to convert to the specified type, usually [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

RETURNS

The value of the argument after casting to float.

SEE ALSO

[int](https://www.tradingview.com/pine-script-reference/v6/#fun_int)[bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool)[color](https://www.tradingview.com/pine-script-reference/v6/#fun_color)[string](https://www.tradingview.com/pine-script-reference/v6/#fun_string)[line](https://www.tradingview.com/pine-script-reference/v6/#fun_line)[label](https://www.tradingview.com/pine-script-reference/v6/#fun_label)

**hline()**

Renders a horizontal line at a given fixed price level.

SYNTAX

hline(price, title, color, linestyle, linewidth, editable, display) → hline

ARGUMENTS

**price (input int/float)** Price value at which the object will be rendered. Required argument.

**title (const string)** Title of the object.

**color (input color)** Color of the rendered line. Must be a constant value (not an expression). Optional argument.

**linestyle (input hline\_style)** Style of the rendered line. Possible values are: [hline.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_hline.style_solid), [hline.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_hline.style_dotted), [hline.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_hline.style_dashed). Optional argument.

**linewidth (input int)** Width of the rendered line. Default value is 1.

**editable (const bool)** If true then hline style will be editable in Format dialog. Default is true.

**display (input plot\_simple\_display)** Controls where the hline is displayed. Possible values are: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

EXAMPLE

//**@version=**6

indicator("input.hline", overlay=true)

hline(3.14, title='Pi', color=color.blue, linestyle=hline.style\_dotted, linewidth=2)

// You may fill the background between any two hlines with a fill() function:

h1 = hline(20)

h2 = hline(10)

fill(h1, h2, color=color.new(color.green, 90))

RETURNS

An hline object, that can be used in [fill](https://www.tradingview.com/pine-script-reference/v6/#fun_fill)

SEE ALSO

[fill](https://www.tradingview.com/pine-script-reference/v6/#fun_fill)

**hour()**2 overloads

SYNTAX & OVERLOADS

[hour(time) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_hour-0)

[hour(time, timezone) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_hour-1)

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

[hour](https://www.tradingview.com/pine-script-reference/v6/#var_hour)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[year](https://www.tradingview.com/pine-script-reference/v6/#fun_year)[month](https://www.tradingview.com/pine-script-reference/v6/#fun_month)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek)[minute](https://www.tradingview.com/pine-script-reference/v6/#fun_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#fun_second)

**indicator()**

This declaration statement designates the script as an indicator and sets a number of indicator-related properties.

SYNTAX

indicator(title, shorttitle, overlay, format, precision, scale, max\_bars\_back, timeframe, timeframe\_gaps, explicit\_plot\_zorder, max\_lines\_count, max\_labels\_count, max\_boxes\_count, calc\_bars\_count, max\_polylines\_count, dynamic\_requests, behind\_chart) → void

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.

**shorttitle (const string)** The script's display name on charts. If specified, it will replace the title argument in most chart-related windows. Optional. The default is the argument used for title.

**overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the indicator will be displayed over the chart. If [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), it will be added in a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**format (const string)** Specifies the formatting of the script's displayed values. Possible values: [format.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_format.inherit), [format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price), [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume), [format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent). Optional. The default is [format.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_format.inherit).

**precision (const int)** Specifies the number of digits after the floating point of the script's displayed values. Must be a non-negative integer no greater than 16. If format is set to [format.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_format.inherit) and precision is specified, the format will instead be set to [format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price). When the function's format parameter uses [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume), the precision parameter will not affect the result, as the decimal precision rules defined by [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume) supersede other precision settings. Optional. The default is inherited from the precision of the chart's symbol.

**scale (const scale\_type)** The price scale used. Possible values: [scale.right](https://www.tradingview.com/pine-script-reference/v6/#const_scale.right), [scale.left](https://www.tradingview.com/pine-script-reference/v6/#const_scale.left), [scale.none](https://www.tradingview.com/pine-script-reference/v6/#const_scale.none). The [scale.none](https://www.tradingview.com/pine-script-reference/v6/#const_scale.none) value can only be applied in combination with overlay = true. Optional. By default, the script uses the same scale as the chart.

**max\_bars\_back (const int)** The length of the historical buffer the script keeps for every variable and function, which determines how many past values can be referenced using the [] history-referencing operator. The required buffer size is automatically detected by the Pine Script® runtime. Using this parameter is only necessary when a runtime error occurs because automatic detection fails. More information on the underlying mechanics of the historical buffer can be found [in our Help Center](https://www.tradingview.com/chart/?solution=43000587849). Optional. The default is 0.

**timeframe (const string)** Adds multi-timeframe functionality to simple scripts. When specified, a "Timeframe" field will be included in the "Calculation" section of the script's "Settings/Inputs" tab. The field's default value will be the argument supplied, whose format must conform to [timeframe string specifications](https://www.tradingview.com/pine-script-docs/concepts/timeframes/#timeframe-string-specifications). To specify the chart's timeframe, use an empty string or the [timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period) variable. The parameter cannot be used with scripts using Pine Script® drawings. Optional. The default is [timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period).

**timeframe\_gaps (const bool)** Specifies how the indicator's values are displayed on chart bars when the timeframe is higher than the chart's. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), a value only appears on a chart bar when the higher timeframe value becomes available, otherwise [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) is returned (thus a "gap" occurs). With [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), what would otherwise be gaps are filled with the latest known value returned, avoiding [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) values. When specified, a "Wait for timeframe closes" checkbox will be included in the "Calculation" section of the script's "Settings/Inputs" tab. Optional. The default is [true](https://www.tradingview.com/pine-script-reference/v6/#const_true).

**explicit\_plot\_zorder (const bool)** Specifies the order in which the script's plots, fills, and hlines are rendered. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), plots are drawn in the order in which they appear in the script's code, each newer plot being drawn above the previous ones. This only applies to plot\*() functions, [fill](https://www.tradingview.com/pine-script-reference/v6/#fun_fill), and [hline](https://www.tradingview.com/pine-script-reference/v6/#fun_hline). Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**max\_lines\_count (const int)** The number of last [line](https://www.tradingview.com/pine-script-reference/v6/#type_line) drawings displayed. Possible values: 1-500. The count is approximate; more drawings than the specified count may be displayed. Optional. The default is 50.

**max\_labels\_count (const int)** The number of last [label](https://www.tradingview.com/pine-script-reference/v6/#type_label) drawings displayed. Possible values: 1-500. The count is approximate; more drawings than the specified count may be displayed. Optional. The default is 50.

**max\_boxes\_count (const int)** The number of last [box](https://www.tradingview.com/pine-script-reference/v6/#type_box) drawings displayed. Possible values: 1-500. The count is approximate; more drawings than the specified count may be displayed. Optional. The default is 50.

**calc\_bars\_count (const int)** Limits the initial calculation of a script to the last number of bars specified. When specified, a "Calculated bars" field will be included in the "Calculation" section of the script's "Settings/Inputs" tab. Optional. The default is 0, in which case the script executes on all available bars.

**max\_polylines\_count (const int)** The number of last [polyline](https://www.tradingview.com/pine-script-reference/v6/#type_polyline) drawings displayed. Possible values: 1-100. The count is approximate; more drawings than the specified count may be displayed. Optional. The default is 50.

**dynamic\_requests (const bool)** Specifies whether the script can dynamically call functions from the request.\*() namespace. Dynamic request.\*() calls are allowed within the local scopes of conditional structures (e.g., [if](https://www.tradingview.com/pine-script-reference/v6/#kw_if)), loops (e.g., [for](https://www.tradingview.com/pine-script-reference/v6/#kw_for)), and exported functions. Additionally, such calls allow "series" arguments for many of their parameters. Optional. The default is [true](https://www.tradingview.com/pine-script-reference/v6/#const_true). See the User Manual's [Dynamic requests](https://www.tradingview.com/pine-script-docs/concepts/other-timeframes-and-data/#dynamic-requests) section for more information.

**behind\_chart (const bool)** Controls whether the script's plots and drawings in the main chart pane appear behind the chart display (if [true](https://www.tradingview.com/pine-script-reference/v6/#const_true)), or in front of it (if [false](https://www.tradingview.com/pine-script-reference/v6/#const_false)). Optional. The default is [true](https://www.tradingview.com/pine-script-reference/v6/#const_true).

EXAMPLE

//**@version=**6

indicator("My script", shorttitle="Script")

plot(close)

REMARKS

Every indicator script must have one [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) call.

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)[library](https://www.tradingview.com/pine-script-reference/v6/#fun_library)

**input()**6 overloads

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

[input(defval, title, tooltip, inline, group, display) → input color](https://www.tradingview.com/pine-script-reference/v6/#fun_input-0)

[input(defval, title, tooltip, inline, group, display) → input string](https://www.tradingview.com/pine-script-reference/v6/#fun_input-1)

[input(defval, title, tooltip, inline, group, display) → input int](https://www.tradingview.com/pine-script-reference/v6/#fun_input-2)

[input(defval, title, tooltip, inline, group, display) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_input-3)

[input(defval, title, inline, group, tooltip, display) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_input-4)

[input(defval, title, tooltip, inline, group, display) → input bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input-5)

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default depends on the type of the value passed to defval: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none) for [bool](https://www.tradingview.com/pine-script-reference/v6/#type_bool) and [color](https://www.tradingview.com/pine-script-reference/v6/#type_color) values, [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all) for everything else.

EXAMPLE

//**@version=**6

indicator("input", overlay=true)

i\_switch = input(true, "On/Off")

plot(i\_switch ? open : na)

i\_len = input(7, "Length")

i\_src = input(close, "Source")

plot(ta.sma(i\_src, i\_len))

i\_border = input(142.50, "Price Border")

hline(i\_border)

bgcolor(close > i\_border ? color.green : color.red)

i\_col = input(color.red, "Plot Color")

plot(close, color=i\_col)

i\_text = input("Hello!", "Message")

l = label.new(bar\_index, high, text=i\_text)

label.delete(l[1])

RETURNS

Value of input variable.

REMARKS

Result of [input](https://www.tradingview.com/pine-script-reference/v6/#fun_input) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_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

input.bool(defval, title, tooltip, inline, group, confirm, display) → input bool

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none).

EXAMPLE

//**@version=**6

indicator("input.bool", overlay=true)

i\_switch = input.bool(true, "On/Off")

plot(i\_switch ? open : na)

RETURNS

Value of input variable.

REMARKS

Result of [input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_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

input.color(defval, title, tooltip, inline, group, confirm, display) → input color

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none).

EXAMPLE

//**@version=**6

indicator("input.color", overlay=true)

i\_col = input.color(color.red, "Plot Color")

plot(close, color=i\_col)

RETURNS

Value of input variable.

REMARKS

Result of [input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#kw_enum) fields passed to its defval and options parameters.

The text for each option in the resulting dropdown corresponds to the titles of the included fields. If a field's title is not specified in the enum declaration, its title is the string representation of its name.

SYNTAX

input.enum(defval, title, options, tooltip, inline, group, confirm, display) → input enum

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**options (tuple of enum fields: [enumName.field1, enumName.field2, ...])** A list of options to choose from. Optional. By default, the titles of all of the enum's fields are available in the dropdown. Passing a tuple as the options argument limits the list to only the included fields.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#var_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#var_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#var_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#var_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#var_display.all).

EXAMPLE

//**@version=**6

indicator("Session highlight", overlay = true)

//**@enum** Contains fields with popular timezones as titles.

//**@field** exch Has an empty string as the title to represent the chart timezone.

enum tz

utc = "UTC"

exch = ""

ny = "America/New\_York"

chi = "America/Chicago"

lon = "Europe/London"

tok = "Asia/Tokyo"

//**@variable** The session string.

selectedSession = input.session("1200-1500", "Session")

//**@variable** The selected timezone. The input's dropdown contains the fields in the `tz` enum.

selectedTimezone = input.enum(tz.utc, "Session Timezone")

//**@variable** Is `true` if the current bar's time is in the specified session.

**bool** inSession = false

if not na(time("", selectedSession, str.tostring(selectedTimezone)))

inSession := true

// Highlight the background when `inSession` is `true`.

bgcolor(inSession ? color.new(color.green, 90) : na, title = "Active session highlight")

RETURNS

Value of input variable.

REMARKS

All fields included in the defval and options arguments must belong to the same enum.

SEE ALSO

[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_input)

**input.float()**2 overloads

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

[input.float(defval, title, options, tooltip, inline, group, confirm, display) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float-0)

[input.float(defval, title, minval, maxval, step, tooltip, inline, group, confirm, display) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float-1)

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**options (tuple of const int/float values: [val1, val2, ...])** A list of options to choose from a dropdown menu, separated by commas and enclosed in square brackets: [val1, val2, ...]. When using this parameter, the minval, maxval and step parameters cannot be used.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

EXAMPLE

//**@version=**6

indicator("input.float", overlay=true)

i\_angle1 = input.float(0.5, "Sin Angle", minval=-3.14, maxval=3.14, step=0.02)

plot(math.sin(i\_angle1) > 0 ? close : open, "sin", color=color.green)

i\_angle2 = input.float(0, "Cos Angle", options=[-3.14, -1.57, 0, 1.57, 3.14])

plot(math.cos(i\_angle2) > 0 ? close : open, "cos", color=color.red)

RETURNS

Value of input variable.

REMARKS

Result of [input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_input)

**input.int()**2 overloads

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

[input.int(defval, title, options, tooltip, inline, group, confirm, display) → input int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int-0)

[input.int(defval, title, minval, maxval, step, tooltip, inline, group, confirm, display) → input int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int-1)

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**options (tuple of const int values: [val1, val2, ...])** A list of options to choose from a dropdown menu, separated by commas and enclosed in square brackets: [val1, val2, ...]. When using this parameter, the minval, maxval and step parameters cannot be used.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

EXAMPLE

//**@version=**6

indicator("input.int", overlay=true)

i\_len1 = input.int(10, "Length 1", minval=5, maxval=21, step=1)

plot(ta.sma(close, i\_len1))

i\_len2 = input.int(10, "Length 2", options=[5, 10, 21])

plot(ta.sma(close, i\_len2))

RETURNS

Value of input variable.

REMARKS

Result of [input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_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

input.price(defval, title, tooltip, inline, group, confirm, display) → input float

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, the interactive input mode is enabled and the selection is done by clicking on the chart when the indicator is added to the chart, or by selecting the indicator and moving the selection after that. Optional. The default is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

EXAMPLE

//**@version=**6

indicator("input.price", overlay=true)

price1 = input.price(title="Date", defval=42)

plot(price1)

price2 = input.price(54, title="Date")

plot(price2)

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

[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.resolution](https://www.tradingview.com/pine-script-reference/v6/#fun_input.resolution)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_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

input.session(defval, title, options, tooltip, inline, group, confirm, display) → input string

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**options (tuple of const string values: [val1, val2, ...])** A list of options to choose from.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

EXAMPLE

//**@version=**6

indicator("input.session", overlay=true)

i\_sess = input.session("1300-1700", "Session", options=["0930-1600", "1300-1700", "1700-2100"])

t = time(timeframe.period, i\_sess)

bgcolor(time == t ? color.green : na)

RETURNS

Value of input variable.

REMARKS

Result of [input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#var_close), [hl2](https://www.tradingview.com/pine-script-reference/v6/#var_hl2), etc. The user can also select an output from another indicator on their chart as the source.

SYNTAX

input.source(defval, title, tooltip, inline, group, display, confirm) → series float

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

EXAMPLE

//**@version=**6

indicator("input.source", overlay=true)

i\_src = input.source(close, "Source")

plot(i\_src)

RETURNS

Value of input variable.

REMARKS

Result of [input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_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

input.string(defval, title, options, tooltip, inline, group, confirm, display) → input string

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**options (tuple of const string values: [val1, val2, ...])** A list of options to choose from.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

EXAMPLE

//**@version=**6

indicator("input.string", overlay=true)

i\_text = input.string("Hello!", "Message")

l = label.new(bar\_index, high, i\_text)

label.delete(l[1])

RETURNS

Value of input variable.

REMARKS

Result of [input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_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

input.symbol(defval, title, tooltip, inline, group, confirm, display) → input string

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

EXAMPLE

//**@version=**6

indicator("input.symbol", overlay=true)

i\_sym = input.symbol("DELL", "Symbol")

s = request.security(i\_sym, 'D', close)

plot(s)

RETURNS

Value of input variable.

REMARKS

Result of [input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_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

input.text\_area(defval, title, tooltip, group, confirm, display) → input string

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none).

EXAMPLE

//**@version=**6

indicator("input.text\_area")

i\_text = input.text\_area(defval = "Hello \nWorld!", title = "Message")

plot(close)

RETURNS

Value of input variable.

REMARKS

Result of [input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_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

input.time(defval, title, tooltip, inline, group, confirm, display) → input int

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](https://www.tradingview.com/pine-script-reference/v6/#fun_timestamp) function, but only if it uses a date argument in const string format.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, the interactive input mode is enabled and the selection is done by clicking on the chart when the indicator is added to the chart, or by selecting the indicator and moving the selection after that. Optional. The default is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none).

EXAMPLE

//**@version=**6

indicator("input.time", overlay=true)

i\_date = input.time(timestamp("20 Jul 2021 00:00 +0300"), "Date")

l = label.new(i\_date, high, "Date", xloc=xloc.bar\_time)

label.delete(l[1])

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

[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_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

input.timeframe(defval, title, options, tooltip, inline, group, confirm, display) → input string

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.

**title (const string)** Title of the input. If not specified, the variable name is used as the input's title. If the title is specified, but it is empty, the name will be an empty string.

**options (tuple of const string values: [val1, val2, ...])** A list of options to choose from.

**tooltip (const string)** The string that will be shown to the user when hovering over the tooltip icon.

**inline (const string)** Combines all the input calls using the same argument in one line. The string used as an argument is not displayed. It is only used to identify inputs belonging to the same line.

**group (const string)** Creates a header above all inputs using the same group argument string. The string is also used as the header's text.

**confirm (const bool)** If true, then user will be asked to confirm input value before indicator is added to chart. Default value is false.

**display (const plot\_display)** Controls where the script will display the input's information, aside from within the script's settings. This option allows one to remove a specific input from the script's status line or the Data Window to ensure only the most necessary inputs are displayed there. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

EXAMPLE

//**@version=**6

indicator("input.timeframe", overlay=true)

i\_res = input.timeframe('D', "Resolution", options=['D', 'W', 'M'])

s = request.security("AAPL", i\_res, close)

plot(s)

RETURNS

Value of input variable.

REMARKS

Result of [input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe) function always should be assigned to a variable, see examples above.

SEE ALSO

[input.bool](https://www.tradingview.com/pine-script-reference/v6/#fun_input.bool)[input.int](https://www.tradingview.com/pine-script-reference/v6/#fun_input.int)[input.float](https://www.tradingview.com/pine-script-reference/v6/#fun_input.float)[input.string](https://www.tradingview.com/pine-script-reference/v6/#fun_input.string)[input.text\_area](https://www.tradingview.com/pine-script-reference/v6/#fun_input.text_area)[input.symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_input.symbol)[input.session](https://www.tradingview.com/pine-script-reference/v6/#fun_input.session)[input.source](https://www.tradingview.com/pine-script-reference/v6/#fun_input.source)[input.color](https://www.tradingview.com/pine-script-reference/v6/#fun_input.color)[input.time](https://www.tradingview.com/pine-script-reference/v6/#fun_input.time)[input](https://www.tradingview.com/pine-script-reference/v6/#fun_input)

**int()**4 overloads

Casts na or truncates float value to int

SYNTAX & OVERLOADS

[int(x) → const int](https://www.tradingview.com/pine-script-reference/v6/#fun_int-0)

[int(x) → input int](https://www.tradingview.com/pine-script-reference/v6/#fun_int-1)

[int(x) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_int-2)

[int(x) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_int-3)

ARGUMENTS

**x (const int/float)** The value to convert to the specified type, usually [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

RETURNS

The value of the argument after casting to int.

SEE ALSO

[float](https://www.tradingview.com/pine-script-reference/v6/#fun_float)[bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool)[color](https://www.tradingview.com/pine-script-reference/v6/#fun_color)[string](https://www.tradingview.com/pine-script-reference/v6/#fun_string)[line](https://www.tradingview.com/pine-script-reference/v6/#fun_line)[label](https://www.tradingview.com/pine-script-reference/v6/#fun_label)

**label()**

Casts na to label

SYNTAX

label(x) → series label

ARGUMENTS

**x (series label)** The value to convert to the specified type, usually [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

RETURNS

The value of the argument after casting to label.

SEE ALSO

[float](https://www.tradingview.com/pine-script-reference/v6/#fun_float)[int](https://www.tradingview.com/pine-script-reference/v6/#fun_int)[bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool)[color](https://www.tradingview.com/pine-script-reference/v6/#fun_color)[string](https://www.tradingview.com/pine-script-reference/v6/#fun_string)[line](https://www.tradingview.com/pine-script-reference/v6/#fun_line)

**label.copy()**

Clones the label object.

SYNTAX

label.copy(id) → series label

ARGUMENTS

**id (series label)** Label object.

EXAMPLE

//**@version=**6

indicator('Last 100 bars highest/lowest', overlay = true)

LOOKBACK = 100

highest = ta.highest(LOOKBACK)

highestBars = ta.highestbars(LOOKBACK)

lowest = ta.lowest(LOOKBACK)

lowestBars = ta.lowestbars(LOOKBACK)

if barstate.islastconfirmedhistory

var labelHigh = label.new(bar\_index + highestBars, highest, str.tostring(highest), color = color.green)

var labelLow = label.copy(labelHigh)

label.set\_xy(labelLow, bar\_index + lowestBars, lowest)

label.set\_text(labelLow, str.tostring(lowest))

label.set\_color(labelLow, color.red)

label.set\_style(labelLow, label.style\_label\_up)

RETURNS

New label ID object which may be passed to label.setXXX and label.getXXX functions.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_label.delete)

**label.delete()**

Deletes the specified label object. If it has already been deleted, does nothing.

SYNTAX

label.delete(id) → void

ARGUMENTS

**id (series label)** Label object to delete.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.get\_text()**

Returns the text of this label object.

SYNTAX

label.get\_text(id) → series string

ARGUMENTS

**id (series label)** Label object.

EXAMPLE

//**@version=**6

indicator("label.get\_text")

my\_label = label.new(time, open, text="Open bar text", xloc=xloc.bar\_time)

a = label.get\_text(my\_label)

label.new(time, close, text = a + " new", xloc=xloc.bar\_time)

RETURNS

String object containing the text of this label.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.get\_x()**

Returns UNIX time or bar index (depending on the last xloc value set) of this label's position.

SYNTAX

label.get\_x(id) → series int

ARGUMENTS

**id (series label)** Label object.

EXAMPLE

//**@version=**6

indicator("label.get\_x")

my\_label = label.new(time, open, text="Open bar text", xloc=xloc.bar\_time)

a = label.get\_x(my\_label)

plot(time - label.get\_x(my\_label)) //draws zero plot

RETURNS

UNIX timestamp (in milliseconds) or bar index.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.get\_y()**

Returns price of this label's position.

SYNTAX

label.get\_y(id) → series float

ARGUMENTS

**id (series label)** Label object.

RETURNS

Floating point value representing price.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.new()**2 overloads

Creates new label object.

SYNTAX & OVERLOADS

[label.new(point, text, xloc, yloc, color, style, textcolor, size, textalign, tooltip, text\_font\_family, force\_overlay, text\_formatting) → series label](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new-0)

[label.new(x, y, text, xloc, yloc, color, style, textcolor, size, textalign, tooltip, text\_font\_family, force\_overlay, text\_formatting) → series label](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new-1)

ARGUMENTS

**point (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object that specifies the label's location.

**text (series string)** Label text. Default is empty string.

**xloc (series string)** See description of **x** argument. Possible values: [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index) and [xloc.bar\_time](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_time). Default is [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index).

**yloc (series string)** Possible values are [yloc.price](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.price), [yloc.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.abovebar), [yloc.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.belowbar). If yloc=[yloc.price](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.price), **y** argument specifies the price of the label position. If yloc=[yloc.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.abovebar), label is located above bar. If yloc=[yloc.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.belowbar), label is located below bar. Default is [yloc.price](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.price).

**color (series color)** Color of the label border and arrow

**style (series string)** Label style. Possible values: [label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none), [label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross), [label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross), [label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup), [label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown), [label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag), [label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle), [label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup), [label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown), [label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up), [label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down), [label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left), [label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right), [label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left), [label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right), [label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left), [label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right), [label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center), [label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square), [label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond), [label.style\_text\_outline](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_text_outline). Default is [label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down).

**textcolor (series color)** Text color.

**size (series int/string)** Optional. Size of the label. Accepts a positive [int](https://www.tradingview.com/pine-script-reference/v6/#type_int) value or one of the built-in size.\* constants. The constants and their equivalent numeric sizes are: [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto) (0), [size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny) (~7), [size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small) (~10), [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal) (12), [size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large) (18), [size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge) (24). The default value is [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal), which represents the numeric size of 12.

**textalign (series string)** Label text alignment. Possible values: [text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left), [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center), [text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_right). Default value is [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center).

**tooltip (series string)** Hover to see tooltip label.

**text\_font\_family (series string)** The font family of the text. Optional. The default value is [font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default). Possible values: [font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default), [font.family\_monospace](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_monospace).

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the drawing will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**text\_formatting (const text\_format)** The formatting of the displayed text. Formatting options support addition. For example, text.format\_bold + text.format\_italic will make the text both bold and italicized. Possible values: [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none), [text.format\_bold](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_bold), [text.format\_italic](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_italic). Optional. The default is [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none).

EXAMPLE

//**@version=**6

indicator("label.new")

var label1 = label.new(bar\_index, low, text="Hello, world!", style=label.style\_circle)

label.set\_x(label1, 0)

label.set\_xloc(label1, time, xloc.bar\_time)

label.set\_color(label1, color.red)

label.set\_size(label1, size.large)

RETURNS

Label ID object which may be passed to label.setXXX and label.getXXX functions.

SEE ALSO

[label.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_label.delete)[label.set\_x](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_x)[label.set\_y](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_y)[label.set\_xy](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_xy)[label.set\_xloc](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_xloc)[label.set\_yloc](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_yloc)[label.set\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_color)[label.set\_textcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textcolor)[label.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_style)[label.set\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_size)[label.set\_textalign](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_textalign)[label.set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_tooltip)[label.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_text)[label.set\_text\_formatting](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_text_formatting)

**label.set\_color()**

Sets label border and arrow color.

SYNTAX

label.set\_color(id, color) → void

ARGUMENTS

**id (series label)** Label object.

**color (series color)** New label border and arrow color.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.set\_point()**

Sets the location of the id label to point.

SYNTAX

label.set\_point(id, point) → void

ARGUMENTS

**id (series label)** A [label](https://www.tradingview.com/pine-script-reference/v6/#type_label) object.

**point (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object.

**label.set\_size()**

Sets arrow and text size of the specified label object.

SYNTAX

label.set\_size(id, size) → void

ARGUMENTS

**id (series label)** Label object.

**size (series int/string)** Size of the label. Accepts a positive [int](https://www.tradingview.com/pine-script-reference/v6/#type_int) value or one of the built-in size.\* constants. The constants and their equivalent numeric sizes are: [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto) (0), [size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny) (~7), [size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small) (~10), [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal) (12), [size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large) (18), [size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge) (24). The default value is [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal), which represents the numeric size of 12.

SEE ALSO

[size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto)[size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny)[size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small)[size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal)[size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large)[size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge)[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.set\_style()**

Sets label style.

SYNTAX

label.set\_style(id, style) → void

ARGUMENTS

**id (series label)** Label object.

**style (series string)** New label style. Possible values: [label.style\_none](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_none), [label.style\_xcross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_xcross), [label.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_cross), [label.style\_triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangleup), [label.style\_triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_triangledown), [label.style\_flag](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_flag), [label.style\_circle](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_circle), [label.style\_arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowup), [label.style\_arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_arrowdown), [label.style\_label\_up](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_up), [label.style\_label\_down](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_down), [label.style\_label\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_left), [label.style\_label\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_right), [label.style\_label\_lower\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_left), [label.style\_label\_lower\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_lower_right), [label.style\_label\_upper\_left](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_left), [label.style\_label\_upper\_right](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_upper_right), [label.style\_label\_center](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_label_center), [label.style\_square](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_square), [label.style\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_diamond), [label.style\_text\_outline](https://www.tradingview.com/pine-script-reference/v6/#const_label.style_text_outline).

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.set\_text()**

Sets label text

SYNTAX

label.set\_text(id, text) → void

ARGUMENTS

**id (series label)** Label object.

**text (series string)** New label text.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_text\_formatting](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_text_formatting)

**label.set\_text\_font\_family()**

The function sets the font family of the text inside the label.

SYNTAX

label.set\_text\_font\_family(id, text\_font\_family) → void

ARGUMENTS

**id (series label)** A label object.

**text\_font\_family (series string)** The font family of the text. Possible values: [font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default), [font.family\_monospace](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_monospace).

EXAMPLE

//**@version=**6

indicator("Example of setting the label font")

if barstate.islastconfirmedhistory

l = label.new(bar\_index, 0, "monospace", yloc=yloc.abovebar)

label.set\_text\_font\_family(l, font.family\_monospace)

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default)[font.family\_monospace](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_monospace)

**label.set\_text\_formatting()**

Sets the formatting attributes the drawing applies to displayed text.

SYNTAX

label.set\_text\_formatting(id, text\_formatting) → void

ARGUMENTS

**id (series label)** Label object.

**text\_formatting (const text\_format)** The formatting of the displayed text. Formatting options support addition. For example, text.format\_bold + text.format\_italic will make the text both bold and italicized. Possible values: [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none), [text.format\_bold](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_bold), [text.format\_italic](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_italic). Optional. The default is [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none).

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)[label.set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_label.set_text)

**label.set\_textalign()**

Sets the alignment for the label text.

SYNTAX

label.set\_textalign(id, textalign) → void

ARGUMENTS

**id (series label)** Label object.

**textalign (series string)** Label text alignment. Possible values: [text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left), [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center), [text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_right).

SEE ALSO

[text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left)[text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center)[text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_right)[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.set\_textcolor()**

Sets color of the label text.

SYNTAX

label.set\_textcolor(id, textcolor) → void

ARGUMENTS

**id (series label)** Label object.

**textcolor (series color)** New text color.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.set\_tooltip()**

Sets the tooltip text.

SYNTAX

label.set\_tooltip(id, tooltip) → void

ARGUMENTS

**id (series label)** Label object.

**tooltip (series string)** Tooltip text.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.set\_x()**

Sets bar index or bar time (depending on the xloc) of the label position.

SYNTAX

label.set\_x(id, x) → void

ARGUMENTS

**id (series label)** Label object.

**x (series int)** New bar index or bar time of the label position. Note that objects positioned using [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index) cannot be drawn further than 500 bars into the future.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.set\_xloc()**

Sets x-location and new bar index/time value.

SYNTAX

label.set\_xloc(id, x, xloc) → void

ARGUMENTS

**id (series label)** Label object.

**x (series int)** New bar index or bar time of the label position.

**xloc (series string)** New x-location value.

SEE ALSO

[xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index)[xloc.bar\_time](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_time)[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.set\_xy()**

Sets bar index/time and price of the label position.

SYNTAX

label.set\_xy(id, x, y) → void

ARGUMENTS

**id (series label)** Label object.

**x (series int)** New bar index or bar time of the label position. Note that objects positioned using [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index) cannot be drawn further than 500 bars into the future.

**y (series int/float)** New price of the label position.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.set\_y()**

Sets price of the label position

SYNTAX

label.set\_y(id, y) → void

ARGUMENTS

**id (series label)** Label object.

**y (series int/float)** New price of the label position.

SEE ALSO

[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**label.set\_yloc()**

Sets new y-location calculation algorithm.

SYNTAX

label.set\_yloc(id, yloc) → void

ARGUMENTS

**id (series label)** Label object.

**yloc (series string)** New y-location value.

SEE ALSO

[yloc.price](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.price)[yloc.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.abovebar)[yloc.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_yloc.belowbar)[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new)

**library()**

Declaration statement identifying a script as a [library](https://www.tradingview.com/pine-script-docs/concepts/libraries/).

SYNTAX

library(title, overlay, dynamic\_requests) → void

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](https://www.tradingview.com/pine-script-reference/v6/#kw_import) statement, when another script uses it. It is also used as the script's name on the chart.

**overlay (const bool)** If true, the library will be added over the chart. If false, it will be added in a separate pane. Optional. The default is false.

**dynamic\_requests (const bool)** Specifies whether the script can dynamically call functions from the request.\*() namespace. Dynamic request.\*() calls are allowed within the local scopes of conditional structures (e.g., [if](https://www.tradingview.com/pine-script-reference/v6/#kw_if)), loops (e.g., [for](https://www.tradingview.com/pine-script-reference/v6/#kw_for)), and exported functions. Additionally, such calls allow "series" arguments for many of their parameters. Optional. The default is [true](https://www.tradingview.com/pine-script-reference/v6/#const_true). See the User Manual's [Dynamic requests](https://www.tradingview.com/pine-script-docs/concepts/other-timeframes-and-data/#dynamic-requests) section for more information.

EXAMPLE

//**@version=**6

// **@description** Math library

library("num\_methods", overlay = true)

// Calculate "sinh()" from the float parameter `x`

export sinh(**float** x) =>

(math.exp(x) - math.exp(-x)) / 2.0

plot(sinh(0))

SEE ALSO

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)

**line()**

Casts na to line

SYNTAX

line(x) → series line

ARGUMENTS

**x (series line)** The value to convert to the specified type, usually [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

RETURNS

The value of the argument after casting to line.

SEE ALSO

[float](https://www.tradingview.com/pine-script-reference/v6/#fun_float)[int](https://www.tradingview.com/pine-script-reference/v6/#fun_int)[bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool)[color](https://www.tradingview.com/pine-script-reference/v6/#fun_color)[string](https://www.tradingview.com/pine-script-reference/v6/#fun_string)[label](https://www.tradingview.com/pine-script-reference/v6/#fun_label)

**line.copy()**

Clones the line object.

SYNTAX

line.copy(id) → series line

ARGUMENTS

**id (series line)** Line object.

EXAMPLE

//**@version=**6

indicator('Last 100 bars price range', overlay = true)

LOOKBACK = 100

highest = ta.highest(LOOKBACK)

lowest = ta.lowest(LOOKBACK)

if barstate.islastconfirmedhistory

var lineTop = line.new(bar\_index[LOOKBACK], highest, bar\_index, highest, color = color.green)

var lineBottom = line.copy(lineTop)

line.set\_y1(lineBottom, lowest)

line.set\_y2(lineBottom, lowest)

line.set\_color(lineBottom, color.red)

RETURNS

New line ID object which may be passed to line.setXXX and line.getXXX functions.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)[line.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_line.delete)

**line.delete()**

Deletes the specified line object. If it has already been deleted, does nothing.

SYNTAX

line.delete(id) → void

ARGUMENTS

**id (series line)** Line object to delete.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.get\_price()**

Returns the price level of a line at a given bar index.

SYNTAX

line.get\_price(id, x) → series float

ARGUMENTS

**id (series line)** Line object.

**x (series int)** Bar index for which price is required.

EXAMPLE

//**@version=**6

indicator("GetPrice", overlay=true)

var **line** l = na

if bar\_index == 10

l := line.new(0, high[5], bar\_index, high)

plot(line.get\_price(l, bar\_index), color=color.green)

RETURNS

Price value of line 'id' at bar index 'x'.

REMARKS

The line is considered to have been created using 'extend=extend.both'.

This function can only be called for lines created using 'xloc.bar\_index'. If you try to call it for a line created with 'xloc.bar\_time', it will generate an error.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_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

line.get\_x1(id) → series int

ARGUMENTS

**id (series line)** Line object.

EXAMPLE

//**@version=**6

indicator("line.get\_x1")

my\_line = line.new(time, open, time + 60 \* 60 \* 24, close, xloc=xloc.bar\_time)

a = line.get\_x1(my\_line)

plot(time - line.get\_x1(my\_line)) //draws zero plot

RETURNS

UNIX timestamp (in milliseconds) or bar index.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_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

line.get\_x2(id) → series int

ARGUMENTS

**id (series line)** Line object.

RETURNS

UNIX timestamp (in milliseconds) or bar index.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.get\_y1()**

Returns price of the first point of the line.

SYNTAX

line.get\_y1(id) → series float

ARGUMENTS

**id (series line)** Line object.

RETURNS

Price value.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.get\_y2()**

Returns price of the second point of the line.

SYNTAX

line.get\_y2(id) → series float

ARGUMENTS

**id (series line)** Line object.

RETURNS

Price value.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.new()**2 overloads

Creates new line object.

SYNTAX & OVERLOADS

[line.new(first\_point, second\_point, xloc, extend, color, style, width, force\_overlay) → series line](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new-0)

[line.new(x1, y1, x2, y2, xloc, extend, color, style, width, force\_overlay) → series line](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new-1)

ARGUMENTS

**first\_point (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object that specifies the line's starting coordinate.

**second\_point (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object that specifies the line's ending coordinate.

**xloc (series string)** See description of **x1** argument. Possible values: [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index) and [xloc.bar\_time](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_time). Default is [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index).

**extend (series string)** If extend=[extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none), draws segment starting at point (x1, y1) and ending at point (x2, y2). If extend is equal to [extend.right](https://www.tradingview.com/pine-script-reference/v6/#const_extend.right) or [extend.left](https://www.tradingview.com/pine-script-reference/v6/#const_extend.left), draws a ray starting at point (x1, y1) or (x2, y2), respectively. If extend=[extend.both](https://www.tradingview.com/pine-script-reference/v6/#const_extend.both), draws a straight line that goes through these points. Default value is [extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none).

**color (series color)** Line color.

**style (series string)** Line style. Possible values: [line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid), [line.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dotted), [line.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dashed), [line.style\_arrow\_left](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_left), [line.style\_arrow\_right](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_right), [line.style\_arrow\_both](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_both).

**width (series int)** Line width in pixels.

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the drawing will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("line.new")

var line1 = line.new(0, low, bar\_index, high, extend=extend.right)

var line2 = line.new(time, open, time + 60 \* 60 \* 24, close, xloc=xloc.bar\_time, style=line.style\_dashed)

line.set\_x2(line1, 0)

line.set\_xloc(line1, time, time + 60 \* 60 \* 24, xloc.bar\_time)

line.set\_color(line2, color.green)

line.set\_width(line2, 5)

RETURNS

Line ID object which may be passed to line.setXXX and line.getXXX functions.

SEE ALSO

[line.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_line.delete)[line.set\_x1](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_x1)[line.set\_y1](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_y1)[line.set\_xy1](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_xy1)[line.set\_x2](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_x2)[line.set\_y2](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_y2)[line.set\_xy2](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_xy2)[line.set\_xloc](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_xloc)[line.set\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_color)[line.set\_extend](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_extend)[line.set\_style](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_style)[line.set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_width)

**line.set\_color()**

Sets the line color

SYNTAX

line.set\_color(id, color) → void

ARGUMENTS

**id (series line)** Line object.

**color (series color)** New line color

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.set\_extend()**

Sets extending type of this line object. If extend=[extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none), draws segment starting at point (x1, y1) and ending at point (x2, y2). If extend is equal to [extend.right](https://www.tradingview.com/pine-script-reference/v6/#const_extend.right) or [extend.left](https://www.tradingview.com/pine-script-reference/v6/#const_extend.left), draws a ray starting at point (x1, y1) or (x2, y2), respectively. If extend=[extend.both](https://www.tradingview.com/pine-script-reference/v6/#const_extend.both), draws a straight line that goes through these points.

SYNTAX

line.set\_extend(id, extend) → void

ARGUMENTS

**id (series line)** Line object.

**extend (series string)** New extending type.

SEE ALSO

[extend.none](https://www.tradingview.com/pine-script-reference/v6/#const_extend.none)[extend.right](https://www.tradingview.com/pine-script-reference/v6/#const_extend.right)[extend.left](https://www.tradingview.com/pine-script-reference/v6/#const_extend.left)[extend.both](https://www.tradingview.com/pine-script-reference/v6/#const_extend.both)[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.set\_first\_point()**

Sets the first point of the id line to point.

SYNTAX

line.set\_first\_point(id, point) → void

ARGUMENTS

**id (series line)** A [line](https://www.tradingview.com/pine-script-reference/v6/#type_line) object.

**point (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object.

**line.set\_second\_point()**

Sets the second point of the id line to point.

SYNTAX

line.set\_second\_point(id, point) → void

ARGUMENTS

**id (series line)** A [line](https://www.tradingview.com/pine-script-reference/v6/#type_line) object.

**point (chart.point)** A [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) object.

**line.set\_style()**

Sets the line style

SYNTAX

line.set\_style(id, style) → void

ARGUMENTS

**id (series line)** Line object.

**style (series string)** New line style.

SEE ALSO

[line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid)[line.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dotted)[line.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dashed)[line.style\_arrow\_left](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_left)[line.style\_arrow\_right](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_right)[line.style\_arrow\_both](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_both)[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.set\_width()**

Sets the line width.

SYNTAX

line.set\_width(id, width) → void

ARGUMENTS

**id (series line)** Line object.

**width (series int)** New line width in pixels.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.set\_x1()**

Sets bar index or bar time (depending on the xloc) of the first point.

SYNTAX

line.set\_x1(id, x) → void

ARGUMENTS

**id (series line)** Line object.

**x (series int)** Bar index or bar time. Note that objects positioned using [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index) cannot be drawn further than 500 bars into the future.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.set\_x2()**

Sets bar index or bar time (depending on the xloc) of the second point.

SYNTAX

line.set\_x2(id, x) → void

ARGUMENTS

**id (series line)** Line object.

**x (series int)** Bar index or bar time. Note that objects positioned using [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index) cannot be drawn further than 500 bars into the future.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.set\_xloc()**

Sets x-location and new bar index/time values.

SYNTAX

line.set\_xloc(id, x1, x2, xloc) → void

ARGUMENTS

**id (series line)** Line object.

**x1 (series int)** Bar index or bar time of the first point.

**x2 (series int)** Bar index or bar time of the second point.

**xloc (series string)** New x-location value.

SEE ALSO

[xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index)[xloc.bar\_time](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_time)[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.set\_xy1()**

Sets bar index/time and price of the first point.

SYNTAX

line.set\_xy1(id, x, y) → void

ARGUMENTS

**id (series line)** Line object.

**x (series int)** Bar index or bar time. Note that objects positioned using [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index) cannot be drawn further than 500 bars into the future.

**y (series int/float)** Price.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.set\_xy2()**

Sets bar index/time and price of the second point

SYNTAX

line.set\_xy2(id, x, y) → void

ARGUMENTS

**id (series line)** Line object.

**x (series int)** Bar index or bar time.

**y (series int/float)** Price.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.set\_y1()**

Sets price of the first point

SYNTAX

line.set\_y1(id, y) → void

ARGUMENTS

**id (series line)** Line object.

**y (series int/float)** Price.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**line.set\_y2()**

Sets price of the second point.

SYNTAX

line.set\_y2(id, y) → void

ARGUMENTS

**id (series line)** Line object.

**y (series int/float)** Price.

SEE ALSO

[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new)

**linefill()**

Casts na to linefill.

SYNTAX

linefill(x) → series linefill

ARGUMENTS

**x (series linefill)** The value to convert to the specified type, usually [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

RETURNS

The value of the argument after casting to linefill.

SEE ALSO

[float](https://www.tradingview.com/pine-script-reference/v6/#fun_float)[int](https://www.tradingview.com/pine-script-reference/v6/#fun_int)[bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool)[color](https://www.tradingview.com/pine-script-reference/v6/#fun_color)[string](https://www.tradingview.com/pine-script-reference/v6/#fun_string)[line](https://www.tradingview.com/pine-script-reference/v6/#fun_line)[label](https://www.tradingview.com/pine-script-reference/v6/#fun_label)

**linefill.delete()**

Deletes the specified linefill object. If it has already been deleted, does nothing.

SYNTAX

linefill.delete(id) → void

ARGUMENTS

**id (series linefill)** A linefill object.

**linefill.get\_line1()**

Returns the ID of the first line used in the id linefill.

SYNTAX

linefill.get\_line1(id) → series line

ARGUMENTS

**id (series linefill)** A linefill object.

**linefill.get\_line2()**

Returns the ID of the second line used in the id linefill.

SYNTAX

linefill.get\_line2(id) → series line

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

linefill.new(line1, line2, color) → series linefill

ARGUMENTS

**line1 (series line)** First line object.

**line2 (series line)** Second line object.

**color (series color)** The color used to fill the space between the lines.

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](https://www.tradingview.com/pine-script-reference/v6/#fun_line.set_xy) functions), the linefill object is also moved.

If both lines are extended in the same direction relative to the lines themselves (e.g. both have [extend.right](https://www.tradingview.com/pine-script-reference/v6/#const_extend.right) as the value of their extend= parameter), the space between line extensions will also be filled.

**linefill.set\_color()**

The function sets the color of the linefill object passed to it.

SYNTAX

linefill.set\_color(id, color) → void

ARGUMENTS

**id (series linefill)** A linefill object.

**color (series color)** The color of the linefill object.

**log.error()**2 overloads

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.

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 function's argument list.

SYNTAX & OVERLOADS

[log.error(message) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_log.error-0)

[log.error(formatString, arg0, arg1, ...) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_log.error-1)

ARGUMENTS

**message (series string)** Log message.

EXAMPLE

//**@version=**6

strategy("My strategy", overlay = true, process\_orders\_on\_close = true)

bracketTickSizeInput = input.int(1000, "Stoploss/Take-Profit distance (in ticks)")

longCondition = ta.crossover(ta.sma(close, 14), ta.sma(close, 28))

if (longCondition)

limitLevel = close \* 1.01

log.info("Long limit order has been placed at {0}", limitLevel)

strategy.order("My Long Entry Id", strategy.long, limit = limitLevel)

log.info("Exit orders have been placed: Take-profit at {0}, Stop-loss at {1}", close, limitLevel)

strategy.exit("Exit", "My Long Entry Id", profit = bracketTickSizeInput, loss = bracketTickSizeInput)

if strategy.opentrades > 10

log.warning("{0} positions opened in the same direction in a row. Try adjusting `bracketTickSizeInput`", strategy.opentrades)

last10Perc = strategy.initial\_capital / 10 > strategy.equity

if (last10Perc and not last10Perc[1])

log.error("The strategy has lost 90% of the initial capital!")

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.

The function can apply additional formatting to some values inside of the {}. The list of additional formatting options can be found in the EXAMPLE section of the [str.format](https://www.tradingview.com/pine-script-reference/v6/#fun_str.format) article.

The string used as the formatString argument can contain single quote characters ('). However, one must pair all single quotes in that string to avoid unexpected formatting results.

The "Pine logs..." button is accessible from the "More" dropdown in the Pine Editor and from the "More" dropdown in the status line of any script that uses log.\*() functions.

**log.info()**2 overloads

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.

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 function's argument list.

SYNTAX & OVERLOADS

[log.info(message) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_log.info-0)

[log.info(formatString, arg0, arg1, ...) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_log.info-1)

ARGUMENTS

**message (series string)** Log message.

EXAMPLE

//**@version=**6

strategy("My strategy", overlay = true, process\_orders\_on\_close = true)

bracketTickSizeInput = input.int(1000, "Stoploss/Take-Profit distance (in ticks)")

longCondition = ta.crossover(ta.sma(close, 14), ta.sma(close, 28))

if (longCondition)

limitLevel = close \* 1.01

log.info("Long limit order has been placed at {0}", limitLevel)

strategy.order("My Long Entry Id", strategy.long, limit = limitLevel)

log.info("Exit orders have been placed: Take-profit at {0}, Stop-loss at {1}", close, limitLevel)

strategy.exit("Exit", "My Long Entry Id", profit = bracketTickSizeInput, loss = bracketTickSizeInput)

if strategy.opentrades > 10

log.warning("{0} positions opened in the same direction in a row. Try adjusting `bracketTickSizeInput`", strategy.opentrades)

last10Perc = strategy.initial\_capital / 10 > strategy.equity

if (last10Perc and not last10Perc[1])

log.error("The strategy has lost 90% of the initial capital!")

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.

The function can apply additional formatting to some values inside of the {}. The list of additional formatting options can be found in the EXAMPLE section of the [str.format](https://www.tradingview.com/pine-script-reference/v6/#fun_str.format) article.

The string used as the formatString argument can contain single quote characters ('). However, one must pair all single quotes in that string to avoid unexpected formatting results.

The "Pine logs..." button is accessible from the "More" dropdown in the Pine Editor and from the "More" dropdown in the status line of any script that uses log.\*() functions.

**log.warning()**2 overloads

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.

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 function's argument list.

SYNTAX & OVERLOADS

[log.warning(message) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_log.warning-0)

[log.warning(formatString, arg0, arg1, ...) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_log.warning-1)

ARGUMENTS

**message (series string)** Log message.

EXAMPLE

//**@version=**6

strategy("My strategy", overlay = true, process\_orders\_on\_close = true)

bracketTickSizeInput = input.int(1000, "Stoploss/Take-Profit distance (in ticks)")

longCondition = ta.crossover(ta.sma(close, 14), ta.sma(close, 28))

if (longCondition)

limitLevel = close \* 1.01

log.info("Long limit order has been placed at {0}", limitLevel)

strategy.order("My Long Entry Id", strategy.long, limit = limitLevel)

log.info("Exit orders have been placed: Take-profit at {0}, Stop-loss at {1}", close, limitLevel)

strategy.exit("Exit", "My Long Entry Id", profit = bracketTickSizeInput, loss = bracketTickSizeInput)

if strategy.opentrades > 10

log.warning("{0} positions opened in the same direction in a row. Try adjusting `bracketTickSizeInput`", strategy.opentrades)

last10Perc = strategy.initial\_capital / 10 > strategy.equity

if (last10Perc and not last10Perc[1])

log.error("The strategy has lost 90% of the initial capital!")

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.

The function can apply additional formatting to some values inside of the {}. The list of additional formatting options can be found in the EXAMPLE section of the [str.format](https://www.tradingview.com/pine-script-reference/v6/#fun_str.format) article.

The string used as the formatString argument can contain single quote characters ('). However, one must pair all single quotes in that string to avoid unexpected formatting results.

The "Pine logs..." button is accessible from the "More" dropdown in the Pine Editor and from the "More" dropdown in the status line of any script that uses log.\*() functions.

**map.clear()**

Clears the map, removing all key-value pairs from it.

SYNTAX

map.clear(id) → void

ARGUMENTS

**id (any map type)** A map object.

EXAMPLE

//**@version=**6

indicator("map.clear example")

oddMap = map.new<**int**, **bool**>()

oddMap.put(1, true)

oddMap.put(2, false)

oddMap.put(3, true)

map.clear(oddMap)

plot(oddMap.size())

SEE ALSO

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)[map.put\_all](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put_all)[map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys)[map.values](https://www.tradingview.com/pine-script-reference/v6/#fun_map.values)[map.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_map.remove)

**map.contains()**

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if the key was found in the id map, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SYNTAX

map.contains(id, key) → series bool

ARGUMENTS

**id (any map type)** A map object.

**key (series <type of the map's elements>)** The key to search in the map.

EXAMPLE

//**@version=**6

indicator("map.includes example")

a = map.new<**string**, **float**>()

a.put("open", open)

p = close

if map.contains(a, "open")

p := a.get("open")

plot(p)

SEE ALSO

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)[map.put](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put)[map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys)[map.values](https://www.tradingview.com/pine-script-reference/v6/#fun_map.values)[map.size](https://www.tradingview.com/pine-script-reference/v6/#fun_map.size)

**map.copy()**

Creates a copy of an existing map.

SYNTAX

map.copy(id) → map<keyType, valueType>

ARGUMENTS

**id (any map type)** A map object to copy.

EXAMPLE

//**@version=**6

indicator("map.copy example")

a = map.new<**string**, **int**>()

a.put("example", 1)

b = map.copy(a)

a := map.new<**string**, **int**>()

a.put("example", 2)

plot(a.get("example"))

plot(b.get("example"))

RETURNS

A copy of the id map.

SEE ALSO

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)[map.put](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put)[map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys)[map.values](https://www.tradingview.com/pine-script-reference/v6/#fun_map.values)[map.get](https://www.tradingview.com/pine-script-reference/v6/#fun_map.get)[map.size](https://www.tradingview.com/pine-script-reference/v6/#fun_map.size)

**map.get()**

Returns the value associated with the specified key in the id map.

SYNTAX

map.get(id, key) → <value\_type>

ARGUMENTS

**id (any map type)** A map object.

**key (series <type of the map's elements>)** The key of the value to retrieve.

EXAMPLE

//**@version=**6

indicator("map.get example")

a = map.new<**int**, **int**>()

size = 10

for i = 0 to size

a.put(i, size-i)

plot(map.get(a, 1))

SEE ALSO

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)[map.put](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put)[map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys)[map.values](https://www.tradingview.com/pine-script-reference/v6/#fun_map.values)[map.contains](https://www.tradingview.com/pine-script-reference/v6/#fun_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

map.keys(id) → array<type>

ARGUMENTS

**id (any map type)** A map object.

EXAMPLE

//**@version=**6

indicator("map.keys example")

a = map.new<**string**, **float**>()

a.put("open", open)

a.put("high", high)

a.put("low", low)

a.put("close", close)

keys = map.keys(a)

ohlc = 0.0

for key in keys

ohlc += a.get(key)

plot(ohlc/4)

REMARKS

Maps maintain insertion order. The elements within the array returned by this function will also be in the insertion order.

SEE ALSO

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)[map.put](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put)[map.get](https://www.tradingview.com/pine-script-reference/v6/#fun_map.get)[map.values](https://www.tradingview.com/pine-script-reference/v6/#fun_map.values)[map.size](https://www.tradingview.com/pine-script-reference/v6/#fun_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.

keyType can be a primitive type or enum. For example: [int](https://www.tradingview.com/pine-script-reference/v6/#type_int), [float](https://www.tradingview.com/pine-script-reference/v6/#type_float), [bool](https://www.tradingview.com/pine-script-reference/v6/#type_bool), [string](https://www.tradingview.com/pine-script-reference/v6/#type_string), [color](https://www.tradingview.com/pine-script-reference/v6/#type_color).

valueType can be of any type except array<>, matrix<>, and map<>. User-defined types are allowed, even if they have array<>, matrix<>, or map<> as one of their fields.

SYNTAX

map.new<keyType, valueType>() → map<keyType, valueType>

EXAMPLE

//**@version=**6

indicator("map.new<string, int> example")

a = map.new<**string**, **int**>()

a.put("example", 1)

label.new(bar\_index, close, str.tostring(a.get("example")))

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.

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

[map.put](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put)[map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys)[map.values](https://www.tradingview.com/pine-script-reference/v6/#fun_map.values)[map.get](https://www.tradingview.com/pine-script-reference/v6/#fun_map.get)[array.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new%3Ctype%3E)

**map.put()**

Puts a new key-value pair into the id map.

SYNTAX

map.put(id, key, value) → <value\_type>

ARGUMENTS

**id (any map type)** A map object.

**key (series <type of the map's elements>)** The key to put into the map.

**value (series <type of the map's elements>)** The key value to put into the map.

EXAMPLE

//**@version=**6

indicator("map.put example")

a = map.new<**string**, **float**>()

map.put(a, "first", 10)

map.put(a, "second", 15)

prevFirst = map.put(a, "first", 20)

currFirst = a.get("first")

plot(prevFirst)

plot(currFirst)

RETURNS

The previous value associated with key if the key was already present in the map, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_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

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)[map.put\_all](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put_all)[map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys)[map.values](https://www.tradingview.com/pine-script-reference/v6/#fun_map.values)[map.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_map.remove)

**map.put\_all()**

Puts all key-value pairs from the id2 map into the id map.

SYNTAX

map.put\_all(id, id2) → void

ARGUMENTS

**id (any map type)** A map object to append to.

**id2 (any map type)** A map object to be appended.

EXAMPLE

//**@version=**6

indicator("map.put\_all example")

a = map.new<**string**, **float**>()

b = map.new<**string**, **float**>()

a.put("first", 10)

a.put("second", 15)

b.put("third", 20)

map.put\_all(a, b)

plot(a.get("third"))

SEE ALSO

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)[map.put](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put)[map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys)[map.values](https://www.tradingview.com/pine-script-reference/v6/#fun_map.values)[map.remove](https://www.tradingview.com/pine-script-reference/v6/#fun_map.remove)

**map.remove()**

Removes a key-value pair from the id map.

SYNTAX

map.remove(id, key) → <value\_type>

ARGUMENTS

**id (any map type)** A map object.

**key (series <type of the map's elements>)** The key of the pair to remove from the map.

EXAMPLE

//**@version=**6

indicator("map.remove example")

a = map.new<**string**, **color**>()

a.put("firstColor", color.green)

oldColorValue = map.remove(a, "firstColor")

plot(close, color = oldColorValue)

RETURNS

The previous value associated with key if the key was present in the map, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) if there was no such key.

SEE ALSO

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)[map.put](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put)[map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys)[map.values](https://www.tradingview.com/pine-script-reference/v6/#fun_map.values)[map.clear](https://www.tradingview.com/pine-script-reference/v6/#fun_map.clear)

**map.size()**

Returns the number of key-value pairs in the id map.

SYNTAX

map.size(id) → series int

ARGUMENTS

**id (any map type)** A map object.

EXAMPLE

//**@version=**6

indicator("map.size example")

a = map.new<**int**, **int**>()

size = 10

for i = 0 to size

a.put(i, size-i)

plot(map.size(a))

SEE ALSO

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)[map.put](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put)[map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys)[map.values](https://www.tradingview.com/pine-script-reference/v6/#fun_map.values)[map.get](https://www.tradingview.com/pine-script-reference/v6/#fun_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

map.values(id) → array<type>

ARGUMENTS

**id (any map type)** A map object.

EXAMPLE

//**@version=**6

indicator("map.values example")

a = map.new<**string**, **float**>()

a.put("open", open)

a.put("high", high)

a.put("low", low)

a.put("close", close)

values = map.values(a)

ohlc = 0.0

for value in values

ohlc += value

plot(ohlc/4)

REMARKS

Maps maintain insertion order. The elements within the array returned by this function will also be in the insertion order.

SEE ALSO

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)[map.put](https://www.tradingview.com/pine-script-reference/v6/#fun_map.put)[map.get](https://www.tradingview.com/pine-script-reference/v6/#fun_map.get)[map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys)[map.size](https://www.tradingview.com/pine-script-reference/v6/#fun_map.size)

**math.abs()**8 overloads

Absolute value of number is number if number >= 0, or -number otherwise.

SYNTAX & OVERLOADS

[math.abs(number) → const int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.abs-0)

[math.abs(number) → input int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.abs-1)

[math.abs(number) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.abs-2)

[math.abs(number) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.abs-3)

[math.abs(number) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.abs-4)

[math.abs(number) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.abs-5)

[math.abs(number) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.abs-6)

[math.abs(number) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.abs-7)

ARGUMENTS

**number (const int)** The number to use in the calculation.

RETURNS

The absolute value of number.

**math.acos()**4 overloads

The acos function returns the arccosine (in radians) of number such that cos(acos(y)) = y for y in range [-1, 1].

SYNTAX & OVERLOADS

[math.acos(angle) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.acos-0)

[math.acos(angle) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.acos-1)

[math.acos(angle) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.acos-2)

[math.acos(angle) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.acos-3)

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](https://www.tradingview.com/pine-script-reference/v6/#var_na) if y is outside of range [-1, 1].

**math.asin()**4 overloads

The asin function returns the arcsine (in radians) of number such that sin(asin(y)) = y for y in range [-1, 1].

SYNTAX & OVERLOADS

[math.asin(angle) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.asin-0)

[math.asin(angle) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.asin-1)

[math.asin(angle) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.asin-2)

[math.asin(angle) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.asin-3)

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](https://www.tradingview.com/pine-script-reference/v6/#var_na) if y is outside of range [-1, 1].

**math.atan()**4 overloads

The atan function returns the arctangent (in radians) of number such that tan(atan(y)) = y for any y.

SYNTAX & OVERLOADS

[math.atan(angle) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.atan-0)

[math.atan(angle) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.atan-1)

[math.atan(angle) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.atan-2)

[math.atan(angle) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.atan-3)

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()**2 overloads

Calculates average of all given series (elementwise).

SYNTAX & OVERLOADS

[math.avg(number0, number1, ...) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.avg-0)

[math.avg(number0, number1, ...) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.avg-1)

ARGUMENTS

**number0, number1, ... (simple int/float)** A sequence of numbers to use in the calculation.

RETURNS

Average.

SEE ALSO

[math.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sum)[ta.cum](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.cum)[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)

**math.ceil()**4 overloads

Rounds the specified number up to the smallest whole number ("int" value) that is greater than or equal to it.

SYNTAX & OVERLOADS

[math.ceil(number) → const int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.ceil-0)

[math.ceil(number) → input int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.ceil-1)

[math.ceil(number) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.ceil-2)

[math.ceil(number) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.ceil-3)

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

[math.floor](https://www.tradingview.com/pine-script-reference/v6/#fun_math.floor)[math.round](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round)

**math.cos()**4 overloads

The cos function returns the trigonometric cosine of an angle.

SYNTAX & OVERLOADS

[math.cos(angle) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.cos-0)

[math.cos(angle) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.cos-1)

[math.cos(angle) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.cos-2)

[math.cos(angle) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.cos-3)

ARGUMENTS

**angle (const int/float)** Angle, in radians.

RETURNS

The trigonometric cosine of an angle.

**math.exp()**4 overloads

The exp function of number is e raised to the power of number, where e is Euler's number.

SYNTAX & OVERLOADS

[math.exp(number) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.exp-0)

[math.exp(number) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.exp-1)

[math.exp(number) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.exp-2)

[math.exp(number) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.exp-3)

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

[math.pow](https://www.tradingview.com/pine-script-reference/v6/#fun_math.pow)

**math.floor()**4 overloads

Rounds the specified number down to the largest whole number ("int" value) that is less than or equal to it.

SYNTAX & OVERLOADS

[math.floor(number) → const int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.floor-0)

[math.floor(number) → input int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.floor-1)

[math.floor(number) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.floor-2)

[math.floor(number) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.floor-3)

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

[math.ceil](https://www.tradingview.com/pine-script-reference/v6/#fun_math.ceil)[math.round](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round)

**math.log()**4 overloads

Natural logarithm of any number > 0 is the unique y such that e^y = number.

SYNTAX & OVERLOADS

[math.log(number) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.log-0)

[math.log(number) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.log-1)

[math.log(number) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.log-2)

[math.log(number) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.log-3)

ARGUMENTS

**number (const int/float)** The number to use in the calculation.

RETURNS

The natural logarithm of number.

SEE ALSO

[math.log10](https://www.tradingview.com/pine-script-reference/v6/#fun_math.log10)

**math.log10()**4 overloads

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

[math.log10(number) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.log10-0)

[math.log10(number) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.log10-1)

[math.log10(number) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.log10-2)

[math.log10(number) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.log10-3)

ARGUMENTS

**number (const int/float)** The number to use in the calculation.

RETURNS

The base 10 logarithm of number.

SEE ALSO

[math.log](https://www.tradingview.com/pine-script-reference/v6/#fun_math.log)

**math.max()**8 overloads

Returns the greatest of multiple values.

SYNTAX & OVERLOADS

[math.max(number0, number1, ...) → const int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.max-0)

[math.max(number0, number1, ...) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.max-1)

[math.max(number0, number1, ...) → input int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.max-2)

[math.max(number0, number1, ...) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.max-3)

[math.max(number0, number1, ...) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.max-4)

[math.max(number0, number1, ...) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.max-5)

[math.max(number0, number1, ...) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.max-6)

[math.max(number0, number1, ...) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.max-7)

ARGUMENTS

**number0, number1, ... (const int)** A sequence of numbers to use in the calculation.

EXAMPLE

//**@version=**6

indicator("math.max", overlay=true)

plot(math.max(close, open))

plot(math.max(close, math.max(open, 42)))

RETURNS

The greatest of multiple given values.

SEE ALSO

[math.min](https://www.tradingview.com/pine-script-reference/v6/#fun_math.min)

**math.min()**8 overloads

Returns the smallest of multiple values.

SYNTAX & OVERLOADS

[math.min(number0, number1, ...) → const int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.min-0)

[math.min(number0, number1, ...) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.min-1)

[math.min(number0, number1, ...) → input int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.min-2)

[math.min(number0, number1, ...) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.min-3)

[math.min(number0, number1, ...) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.min-4)

[math.min(number0, number1, ...) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.min-5)

[math.min(number0, number1, ...) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.min-6)

[math.min(number0, number1, ...) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.min-7)

ARGUMENTS

**number0, number1, ... (const int)** A sequence of numbers to use in the calculation.

EXAMPLE

//**@version=**6

indicator("math.min", overlay=true)

plot(math.min(close, open))

plot(math.min(close, math.min(open, 42)))

RETURNS

The smallest of multiple given values.

SEE ALSO

[math.max](https://www.tradingview.com/pine-script-reference/v6/#fun_math.max)

**math.pow()**4 overloads

Mathematical power function.

SYNTAX & OVERLOADS

[math.pow(base, exponent) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.pow-0)

[math.pow(base, exponent) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.pow-1)

[math.pow(base, exponent) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.pow-2)

[math.pow(base, exponent) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.pow-3)

ARGUMENTS

**base (const int/float)** Specify the base to use.

**exponent (const int/float)** Specifies the exponent.

EXAMPLE

//**@version=**6

indicator("math.pow", overlay=true)

plot(math.pow(close, 2))

RETURNS

base raised to the power of exponent. If base is a series, it is calculated elementwise.

SEE ALSO

[math.sqrt](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sqrt)[math.exp](https://www.tradingview.com/pine-script-reference/v6/#fun_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

math.random(min, max, seed) → series float

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.

**max (series int/float)** The upper bound of the range of random values. The value is not included in the range. The default is 1.

**seed (series int)** Optional argument. When the same seed is used, allows successive calls to the function to produce a repeatable set of values.

RETURNS

A random value.

**math.round()**8 overloads

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

[math.round(number) → const int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round-0)

[math.round(number) → input int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round-1)

[math.round(number) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round-2)

[math.round(number) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round-3)

[math.round(number, precision) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round-4)

[math.round(number, precision) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round-5)

[math.round(number, precision) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round-6)

[math.round(number, precision) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round-7)

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

[math.ceil](https://www.tradingview.com/pine-script-reference/v6/#fun_math.ceil)[math.floor](https://www.tradingview.com/pine-script-reference/v6/#fun_math.floor)

**math.round\_to\_mintick()**2 overloads

Returns the value rounded to the symbol's mintick, i.e. the nearest value that can be divided by [syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mintick), without the remainder, with ties rounding up.

SYNTAX & OVERLOADS

[math.round\_to\_mintick(number) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round_to_mintick-0)

[math.round\_to\_mintick(number) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.round_to_mintick-1)

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

[math.ceil](https://www.tradingview.com/pine-script-reference/v6/#fun_math.ceil)[math.floor](https://www.tradingview.com/pine-script-reference/v6/#fun_math.floor)

**math.sign()**4 overloads

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

[math.sign(number) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sign-0)

[math.sign(number) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sign-1)

[math.sign(number) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sign-2)

[math.sign(number) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sign-3)

ARGUMENTS

**number (const int/float)** The number to use in the calculation.

RETURNS

The sign of the argument.

**math.sin()**4 overloads

The sin function returns the trigonometric sine of an angle.

SYNTAX & OVERLOADS

[math.sin(angle) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sin-0)

[math.sin(angle) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sin-1)

[math.sin(angle) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sin-2)

[math.sin(angle) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sin-3)

ARGUMENTS

**angle (const int/float)** Angle, in radians.

RETURNS

The trigonometric sine of an angle.

**math.sqrt()**4 overloads

Square root of any number >= 0 is the unique y >= 0 such that y^2 = number.

SYNTAX & OVERLOADS

[math.sqrt(number) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sqrt-0)

[math.sqrt(number) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sqrt-1)

[math.sqrt(number) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sqrt-2)

[math.sqrt(number) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sqrt-3)

ARGUMENTS

**number (const int/float)** The number to use in the calculation.

RETURNS

The square root of number.

SEE ALSO

[math.pow](https://www.tradingview.com/pine-script-reference/v6/#fun_math.pow)

**math.sum()**

The sum function returns the sliding sum of last y values of x.

SYNTAX

math.sum(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

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

[ta.cum](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.cum)[for](https://www.tradingview.com/pine-script-reference/v6/#kw_for)

**math.tan()**4 overloads

The tan function returns the trigonometric tangent of an angle.

SYNTAX & OVERLOADS

[math.tan(angle) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.tan-0)

[math.tan(angle) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.tan-1)

[math.tan(angle) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.tan-2)

[math.tan(angle) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_math.tan-3)

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

math.todegrees(radians) → series float

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

math.toradians(degrees) → series float

ARGUMENTS

**degrees (series int/float)** Angle in degrees.

RETURNS

The angle value in radians.

**matrix.add\_col()**2 overloads

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

[matrix.add\_col(id, column) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.add_col-0)

[matrix.add\_col(id, column, array\_id) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.add_col-1)

ARGUMENTS

**id (any matrix type)** A matrix object.

**column (series int)** The index of the column after which the new column will be inserted. Optional. The default value is [matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns).

Adding a column to the matrix

EXAMPLE

//**@version=**6

indicator("`matrix.add\_col()` Example 1")

// Create a 2x3 "int" matrix containing values `0`.

m = matrix.new<**int**>(2, 3, 0)

// Add a column with `na` values to the matrix.

matrix.add\_col(m)

// Display matrix elements.

if barstate.islastconfirmedhistory

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Matrix elements:")

table.cell(t, 0, 1, str.tostring(m))

Adding an array as a column to the matrix

EXAMPLE

//**@version=**6

indicator("`matrix.add\_col()` Example 2")

if barstate.islastconfirmedhistory

// Create an empty matrix object.

var m = matrix.new<**int**>()

// Create an array with values `1` and `3`.

var a = array.from(1, 3)

// Add the `a` array as the first column of the empty matrix.

matrix.add\_col(m, 0, a)

// Display matrix elements.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Matrix elements:")

table.cell(t, 0, 1, str.tostring(m))

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](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.add_row) function.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)[matrix.add\_row](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.add_row)

**matrix.add\_row()**2 overloads

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

[matrix.add\_row(id, row) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.add_row-0)

[matrix.add\_row(id, row, array\_id) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.add_row-1)

ARGUMENTS

**id (any matrix type)** A matrix object.

**row (series int)** The index of the row after which the new row will be inserted. Optional. The default value is [matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows).

Adding a row to the matrix

EXAMPLE

//**@version=**6

indicator("`matrix.add\_row()` Example 1")

// Create a 2x3 "int" matrix containing values `0`.

m = matrix.new<**int**>(2, 3, 0)

// Add a row with `na` values to the matrix.

matrix.add\_row(m)

// Display matrix elements.

if barstate.islastconfirmedhistory

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Matrix elements:")

table.cell(t, 0, 1, str.tostring(m))

Adding an array as a row to the matrix

EXAMPLE

//**@version=**6

indicator("`matrix.add\_row()` Example 2")

if barstate.islastconfirmedhistory

// Create an empty matrix object.

var m = matrix.new<**int**>()

// Create an array with values `1` and `2`.

var a = array.from(1, 2)

// Add the `a` array as the first row of the empty matrix.

matrix.add\_row(m, 0, a)

// Display matrix elements.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Matrix elements:")

table.cell(t, 0, 1, str.tostring(m))

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)[matrix.add\_col](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.add_col)

**matrix.avg()**2 overloads

The function calculates the average of all elements in the matrix.

SYNTAX & OVERLOADS

[matrix.avg(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.avg-0)

[matrix.avg(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.avg-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.avg()` Example")

// Create a 2x2 matrix.

var m = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m, 0, 0, 1)

matrix.set(m, 0, 1, 2)

matrix.set(m, 1, 0, 3)

matrix.set(m, 1, 1, 4)

// Get the average value of the matrix.

var x = matrix.avg(m)

plot(x, 'Matrix average value')

RETURNS

The average value from the id matrix.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.col()**

The function creates a one-dimensional array from the elements of a matrix column.

SYNTAX

matrix.col(id, column) → array<type>

ARGUMENTS

**id (any matrix type)** A matrix object.

**column (series int)** Index of the required column.

EXAMPLE

//**@version=**6

indicator("`matrix.col()` Example", "", true)

// Create a 2x3 "float" matrix from `hlc3` values.

m = matrix.new<**float**>(2, 3, hlc3)

// Return an array with the values of the first column of matrix `m`.

a = matrix.col(m, 0)

// Plot the first value from the array `a`.

plot(array.get(a, 0))

RETURNS

An array ID containing the column values of the id matrix.

REMARKS

Indexing of rows starts at 0.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[matrix.col](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.col)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)

**matrix.columns()**

The function returns the number of columns in the matrix.

SYNTAX

matrix.columns(id) → series int

ARGUMENTS

**id (any matrix type)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.columns()` Example")

// Create a 2x6 matrix with values `0`.

var m = matrix.new<**int**>(2, 6, 0)

// Get the quantity of columns in matrix `m`.

var x = matrix.columns(m)

// Display using a label.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, "Columns: " + str.tostring(x) + "\n" + str.tostring(m))

RETURNS

The number of columns in the matrix id.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.col](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.col)[matrix.row](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.row)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.concat()**

The function appends the m2 matrix to the m1 matrix.

SYNTAX

matrix.concat(id1, id2) → matrix<type>

ARGUMENTS

**id1 (any matrix type)** Matrix object to concatenate into.

**id2 (any matrix type)** Matrix object whose elements will be appended to id1.

EXAMPLE

//**@version=**6

indicator("`matrix.concat()` Example")

// Create a 2x4 "int" matrix containing values `0`.

m1 = matrix.new<**int**>(2, 4, 0)

// Create a 2x4 "int" matrix containing values `1`.

m2 = matrix.new<**int**>(2, 4, 1)

// Append matrix `m2` to `m1`.

matrix.concat(m1, m2)

// Display matrix elements.

if barstate.islastconfirmedhistory

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Matrix Elements:")

table.cell(t, 0, 1, str.tostring(m1))

RETURNS

Returns the id1 matrix concatenated with the id2 matrix.

REMARKS

The number of columns in both matrices must be identical.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.copy()**

The function creates a new matrix which is a copy of the original.

SYNTAX

matrix.copy(id) → matrix<type>

ARGUMENTS

**id (any matrix type)** A matrix object to copy.

EXAMPLE

//**@version=**6

indicator("`matrix.copy()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x3 "float" matrix with `1` values.

var m1 = matrix.new<**float**>(2, 3, 1)

// Copy the matrix to a new one.

// Note that unlike what `matrix.copy()` does,

// the simple assignment operation `m2 = m1`

// would NOT create a new copy of the `m1` matrix.

// It would merely create a copy of its ID referencing the same matrix.

var m2 = matrix.copy(m1)

// Display using a table.

var t = table.new(position.top\_right, 5, 2, color.green)

table.cell(t, 0, 0, "Original Matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Matrix Copy:")

table.cell(t, 1, 1, str.tostring(m2))

RETURNS

A new matrix object of the copied id matrix.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.det()**2 overloads

The function returns the [determinant](https://en.wikipedia.org/wiki/Determinant) of a square matrix.

SYNTAX & OVERLOADS

[matrix.det(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.det-0)

[matrix.det(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.det-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.det` Example")

// Create a 2x2 matrix.

var m = matrix.new<**float**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m, 0, 0, 3)

matrix.set(m, 0, 1, 7)

matrix.set(m, 1, 0, 1)

matrix.set(m, 1, 1, -4)

// Get the determinant of the matrix.

var x = matrix.det(m)

plot(x, 'Matrix determinant')

RETURNS

The determinant value of the id matrix.

REMARKS

Function calculation based on the [LU decomposition](https://en.wikipedia.org/wiki/LU_decomposition) algorithm.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.is\_square](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_square)

**matrix.diff()**2 overloads

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

[matrix.diff(id1, id2) → matrix<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.diff-0)

[matrix.diff(id1, id2) → matrix<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.diff-1)

ARGUMENTS

**id1 (matrix<int>)** Matrix to subtract from.

**id2 (series int/float/matrix<int>)** Matrix object or a scalar value to be subtracted.

Difference between two matrices

EXAMPLE

//**@version=**6

indicator("`matrix.diff()` Example 1")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x3 matrix containing values `5`.

var m1 = matrix.new<**float**>(2, 3, 5)

// Create a 2x3 matrix containing values `4`.

var m2 = matrix.new<**float**>(2, 3, 4)

// Create a new matrix containing the difference between matrices `m1` and `m2`.

var m3 = matrix.diff(m1, m2)

// Display using a table.

var t = table.new(position.top\_right, 1, 2, color.green)

table.cell(t, 0, 0, "Difference between two matrices:")

table.cell(t, 0, 1, str.tostring(m3))

Difference between a matrix and a scalar value

EXAMPLE

//**@version=**6

indicator("`matrix.diff()` Example 2")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x3 matrix with values `4`.

var m1 = matrix.new<**float**>(2, 3, 4)

// Create a new matrix containing the difference between the `m1` matrix and the "int" value `1`.

var m2 = matrix.diff(m1, 1)

// Display using a table.

var t = table.new(position.top\_right, 1, 2, color.green)

table.cell(t, 0, 0, "Difference between a matrix and a scalar:")

table.cell(t, 0, 1, str.tostring(m2))

RETURNS

A new matrix object containing the difference between id2 and id1.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.eigenvalues()**2 overloads

The function returns an array containing the [eigenvalues](https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors) of a square matrix.

SYNTAX & OVERLOADS

[matrix.eigenvalues(id) → array<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.eigenvalues-0)

[matrix.eigenvalues(id) → array<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.eigenvalues-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.eigenvalues()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x2 matrix.

var m1 = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 2)

matrix.set(m1, 0, 1, 4)

matrix.set(m1, 1, 0, 6)

matrix.set(m1, 1, 1, 8)

// Get the eigenvalues of the matrix.

tr = matrix.eigenvalues(m1)

// Display matrix elements.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Matrix elements:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Array of Eigenvalues:")

table.cell(t, 1, 1, str.tostring(tr))

RETURNS

An array containing the eigenvalues of the id matrix.

REMARKS

The function is calculated using "The Implicit QL Algorithm".

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.eigenvectors](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.eigenvectors)

**matrix.eigenvectors()**2 overloads

Returns a matrix of [eigenvectors](https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors), in which each column is an eigenvector of the id matrix.

SYNTAX & OVERLOADS

[matrix.eigenvectors(id) → matrix<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.eigenvectors-0)

[matrix.eigenvectors(id) → matrix<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.eigenvectors-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.eigenvectors()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x2 matrix

var m1 = matrix.new<**int**>(2, 2, 1)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 2)

matrix.set(m1, 0, 1, 4)

matrix.set(m1, 1, 0, 6)

matrix.set(m1, 1, 1, 8)

// Get the eigenvectors of the matrix.

m2 = matrix.eigenvectors(m1)

// Display matrix elements.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Matrix Elements:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Matrix Eigenvectors:")

table.cell(t, 1, 1, str.tostring(m2))

RETURNS

A new matrix containing the eigenvectors of the id matrix.

REMARKS

The function is calculated using "The Implicit QL Algorithm".

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.eigenvalues](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.eigenvalues)

**matrix.elements\_count()**

The function returns the total number of all matrix elements.

SYNTAX

matrix.elements\_count(id) → series int

ARGUMENTS

**id (any matrix type)** A matrix object.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_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

matrix.fill(id, value, from\_row, to\_row, from\_column, to\_column) → void

ARGUMENTS

**id (any matrix type)** A matrix object.

**value (series <type of the matrix's elements>)** The value to fill with.

**from\_row (series int)** Row index from which the fill will begin (inclusive). Optional. The default value is 0.

**to\_row (series int)** Row index where the fill will end (not inclusive). Optional. The default value is [matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows).

**from\_column (series int)** Column index from which the fill will begin (inclusive). Optional. The default value is 0.

**to\_column (series int)** Column index where the fill will end (non inclusive). Optional. The default value is [matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns).

EXAMPLE

//**@version=**6

indicator("`matrix.fill()` Example")

// Create a 4x5 "int" matrix containing values `0`.

m = matrix.new<**float**>(4, 5, 0)

// Fill the intersection of rows 1 to 2 and columns 2 to 3 of the matrix with `hl2` values.

matrix.fill(m, hl2, 0, 2, 1, 3)

// Display using a label.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, str.tostring(m))

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.get()**

The function returns the element with the specified index of the matrix.

SYNTAX

matrix.get(id, row, column) → <matrix\_type>

ARGUMENTS

**id (any matrix type)** A matrix object.

**row (series int)** Index of the required row.

**column (series int)** Index of the required column.

EXAMPLE

//**@version=**6

indicator("`matrix.get()` Example", "", true)

// Create a 2x3 "float" matrix from the `hl2` values.

m = matrix.new<**float**>(2, 3, hl2)

// Return the value of the element at index [0, 0] of matrix `m`.

x = matrix.get(m, 0, 0)

plot(x)

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.inv()**2 overloads

The function returns the [inverse](https://en.wikipedia.org/wiki/Invertible_matrix) of a square matrix.

SYNTAX & OVERLOADS

[matrix.inv(id) → matrix<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.inv-0)

[matrix.inv(id) → matrix<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.inv-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.inv()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x2 matrix.

var m1 = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 1)

matrix.set(m1, 0, 1, 2)

matrix.set(m1, 1, 0, 3)

matrix.set(m1, 1, 1, 4)

// Inverse of the matrix.

var m2 = matrix.inv(m1)

// Display matrix elements.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Original Matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Inverse matrix:")

table.cell(t, 1, 1, str.tostring(m2))

RETURNS

A new matrix, which is the inverse of the id matrix.

REMARKS

The function is calculated using the [LU decomposition](https://en.wikipedia.org/wiki/LU_decomposition) algorithm.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.pinv](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.pinv)[matrix.copy](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.copy)[str.tostring](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tostring)

**matrix.is\_antidiagonal()**

The function determines if the matrix is [anti-diagonal](https://en.wikipedia.org/wiki/Anti-diagonal_matrix) (all elements outside the secondary diagonal are zero).

SYNTAX

matrix.is\_antidiagonal(id) → series bool

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.is\_square](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_square)[matrix.is\_identity](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_identity)[matrix.is\_diagonal](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_diagonal)

**matrix.is\_antisymmetric()**

The function determines if a matrix is [antisymmetric](https://en.wikipedia.org/wiki/Skew-symmetric_matrix) (its [transpose](https://en.wikipedia.org/wiki/Transpose) equals its negative).

SYNTAX

matrix.is\_antisymmetric(id) → series bool

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.is\_square](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_square)

**matrix.is\_binary()**

The function determines if the matrix is [binary](https://en.wikipedia.org/wiki/Logical_matrix) (when all elements of the matrix are 0 or 1).

SYNTAX

matrix.is\_binary(id) → series bool

ARGUMENTS

**id (matrix<int/float>)** Matrix object to test.

RETURNS

Returns true if the id matrix is binary, false otherwise.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)

**matrix.is\_diagonal()**

The function determines if the matrix is [diagonal](https://en.wikipedia.org/wiki/Diagonal_matrix) (all elements outside the main diagonal are zero).

SYNTAX

matrix.is\_diagonal(id) → series bool

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.is\_square](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_square)[matrix.is\_identity](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_identity)[matrix.is\_antidiagonal](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_antidiagonal)

**matrix.is\_identity()**

The function determines if a matrix is an [identity matrix](https://en.wikipedia.org/wiki/Identity_matrix) (elements with ones on the [main diagonal](https://en.wikipedia.org/wiki/Main_diagonal) and zeros elsewhere).

SYNTAX

matrix.is\_identity(id) → series bool

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.is\_square](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_square)[matrix.is\_diagonal](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_diagonal)

**matrix.is\_square()**

The function determines if the matrix is [square](https://en.wikipedia.org/wiki/Square_matrix) (it has the same number of rows and columns).

SYNTAX

matrix.is\_square(id) → series bool

ARGUMENTS

**id (any matrix type)** Matrix object to test.

RETURNS

Returns true if the id matrix is square, false otherwise.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.is\_stochastic()**

The function determines if the matrix is [stochastic](https://en.wikipedia.org/wiki/Stochastic_matrix).

SYNTAX

matrix.is\_stochastic(id) → series bool

ARGUMENTS

**id (matrix<int/float>)** Matrix object to test.

RETURNS

Returns true if the id matrix is stochastic, false otherwise.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)

**matrix.is\_symmetric()**

The function determines if a [square matrix](https://en.wikipedia.org/wiki/Square_matrix) is [symmetric](https://en.wikipedia.org/wiki/Symmetric_matrix) (elements are symmetric with respect to the [main diagonal](https://en.wikipedia.org/wiki/Main_diagonal)).

SYNTAX

matrix.is\_symmetric(id) → series bool

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.is\_square](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_square)

**matrix.is\_triangular()**

The function determines if the matrix is [triangular](https://en.wikipedia.org/wiki/Triangular_matrix) (if all elements above or below the [main diagonal](https://en.wikipedia.org/wiki/Main_diagonal) are zero).

SYNTAX

matrix.is\_triangular(id) → series bool

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.is\_square](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.is_square)

**matrix.is\_zero()**

The function determines if all elements of the matrix are zero.

SYNTAX

matrix.is\_zero(id) → series bool

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)

**matrix.kron()**2 overloads

The function returns the [Kronecker product](https://en.wikipedia.org/wiki/Kronecker_product) for the id1 and id2 matrices.

SYNTAX & OVERLOADS

[matrix.kron(id1, id2) → matrix<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.kron-0)

[matrix.kron(id1, id2) → matrix<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.kron-1)

ARGUMENTS

**id1 (matrix<int/float>)** First matrix object.

**id2 (matrix<int/float>)** Second matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.kron()` Example")

// Display using a table.

if barstate.islastconfirmedhistory

// Create two matrices with default values `1` and `2`.

var m1 = matrix.new<**float**>(2, 2, 1)

var m2 = matrix.new<**float**>(2, 2, 2)

// Calculate the Kronecker product of the matrices.

var m3 = matrix.kron(m1, m2)

// Display matrix elements.

var t = table.new(position.top\_right, 5, 2, color.green)

table.cell(t, 0, 0, "Matrix 1:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 1, "⊗")

table.cell(t, 2, 0, "Matrix 2:")

table.cell(t, 2, 1, str.tostring(m2))

table.cell(t, 3, 1, "=")

table.cell(t, 4, 0, "Kronecker product:")

table.cell(t, 4, 1, str.tostring(m3))

RETURNS

A new matrix containing the [Kronecker product](https://en.wikipedia.org/wiki/Kronecker_product) of id1 and id2.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.mult](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.mult)[str.tostring](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tostring)[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)

**matrix.max()**2 overloads

The function returns the largest value from the matrix elements.

SYNTAX & OVERLOADS

[matrix.max(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.max-0)

[matrix.max(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.max-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.max()` Example")

// Create a 2x2 matrix.

var m = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m, 0, 0, 1)

matrix.set(m, 0, 1, 2)

matrix.set(m, 1, 0, 3)

matrix.set(m, 1, 1, 4)

// Get the maximum value in the matrix.

var x = matrix.max(m)

plot(x, 'Matrix maximum value')

RETURNS

The maximum value from the id matrix.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.min](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.min)[matrix.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.avg)[matrix.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.sort)

**matrix.median()**2 overloads

The function calculates the [median](https://en.wikipedia.org/wiki/Median) ("the middle" value) of matrix elements.

SYNTAX & OVERLOADS

[matrix.median(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.median-0)

[matrix.median(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.median-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.median()` Example")

// Create a 2x2 matrix.

m = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m, 0, 0, 1)

matrix.set(m, 0, 1, 2)

matrix.set(m, 1, 0, 3)

matrix.set(m, 1, 1, 4)

// Get the median of the matrix.

x = matrix.median(m)

plot(x, 'Median of the matrix')

REMARKS

Note that [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) elements of the matrix are not considered when calculating the median.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.mode](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.mode)[matrix.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.sort)[matrix.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.avg)

**matrix.min()**2 overloads

The function returns the smallest value from the matrix elements.

SYNTAX & OVERLOADS

[matrix.min(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.min-0)

[matrix.min(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.min-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.min()` Example")

// Create a 2x2 matrix.

var m = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m, 0, 0, 1)

matrix.set(m, 0, 1, 2)

matrix.set(m, 1, 0, 3)

matrix.set(m, 1, 1, 4)

// Get the minimum value from the matrix.

var x = matrix.min(m)

plot(x, 'Matrix minimum value')

RETURNS

The smallest value from the id matrix.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.max](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.max)[matrix.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.avg)[matrix.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.sort)

**matrix.mode()**2 overloads

The function calculates the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) 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

[matrix.mode(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.mode-0)

[matrix.mode(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.mode-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.mode()` Example")

// Create a 2x2 matrix.

var m = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m, 0, 0, 0)

matrix.set(m, 0, 1, 0)

matrix.set(m, 1, 0, 1)

matrix.set(m, 1, 1, 1)

// Get the mode of the matrix.

var x = matrix.mode(m)

plot(x, 'Mode of the matrix')

RETURNS

The most frequently occurring value from the id matrix. If none exists, returns the smallest value instead.

REMARKS

Note that [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) elements of the matrix are not considered when calculating the mode.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.median](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.median)[matrix.sort](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.sort)[matrix.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.avg)

**matrix.mult()**4 overloads

The function returns a new matrix resulting from the [product](https://en.wikipedia.org/wiki/Matrix_multiplication) 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

[matrix.mult(id1, id2) → array<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.mult-0)

[matrix.mult(id1, id2) → array<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.mult-1)

[matrix.mult(id1, id2) → matrix<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.mult-2)

[matrix.mult(id1, id2) → matrix<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.mult-3)

ARGUMENTS

**id1 (matrix<int>)** First matrix object.

**id2 (array<int>)** Second matrix object, value or array.

Product of two matrices

EXAMPLE

//**@version=**6

indicator("`matrix.mult()` Example 1")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 6x2 matrix containing values `5`.

var m1 = matrix.new<**float**>(6, 2, 5)

// Create a 2x3 matrix containing values `4`.

// Note that it must have the same quantity of rows as there are columns in the first matrix.

var m2 = matrix.new<**float**>(2, 3, 4)

// Create a new matrix from the multiplication of the two matrices.

var m3 = matrix.mult(m1, m2)

// Display using a table.

var t = table.new(position.top\_right, 1, 2, color.green)

table.cell(t, 0, 0, "Product of two matrices:")

table.cell(t, 0, 1, str.tostring(m3))

Product of a matrix and a scalar

EXAMPLE

//**@version=**6

indicator("`matrix.mult()` Example 2")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x3 matrix containing values `4`.

var m1 = matrix.new<**float**>(2, 3, 4)

// Create a new matrix from the product of the two matrices.

scalar = 5

var m2 = matrix.mult(m1, scalar)

// Display using a table.

var t = table.new(position.top\_right, 5, 2, color.green)

table.cell(t, 0, 0, "Matrix 1:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 1, "x")

table.cell(t, 2, 0, "Scalar:")

table.cell(t, 2, 1, str.tostring(scalar))

table.cell(t, 3, 1, "=")

table.cell(t, 4, 0, "Matrix 2:")

table.cell(t, 4, 1, str.tostring(m2))

Product of a matrix and an array vector

EXAMPLE

//**@version=**6

indicator("`matrix.mult()` Example 3")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x3 matrix containing values `4`.

var m1 = matrix.new<**int**>(2, 3, 4)

// Create an array of three elements.

var **int**[] a = array.from(1, 1, 1)

// Create a new matrix containing the product of the `m1` matrix and the `a` array.

var m3 = matrix.mult(m1, a)

// Display using a table.

var t = table.new(position.top\_right, 5, 2, color.green)

table.cell(t, 0, 0, "Matrix 1:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 1, "x")

table.cell(t, 2, 0, "Value:")

table.cell(t, 2, 1, str.tostring(a, " "))

table.cell(t, 3, 1, "=")

table.cell(t, 4, 0, "Matrix 3:")

table.cell(t, 4, 1, str.tostring(m3))

RETURNS

A new matrix object containing the product of id2 and id1.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.sum)[matrix.diff](https://www.tradingview.com/pine-script-reference/v6/#fun_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

matrix.new<type>(rows, columns, initial\_value) → matrix<type>

ARGUMENTS

**rows (series int)** Initial row count of the matrix. Optional. The default value is 0.

**columns (series int)** Initial column count of the matrix. Optional. The default value is 0.

**initial\_value (<matrix\_type>)** Initial value of all matrix elements. Optional. The default is 'na'.

Create a matrix of elements with the same initial value

EXAMPLE

//**@version=**6

indicator("`matrix.new<type>()` Example 1")

// Create a 2x3 (2 rows x 3 columns) "int" matrix with values zero.

var m = matrix.new<**int**>(2, 3, 0)

// Display using a label.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, str.tostring(m))

Create a matrix from array values

EXAMPLE

//**@version=**6

indicator("`matrix.new<type>()` Example 2")

// Function to create a matrix whose rows are filled with array values.

matrixFromArray(**int** rows, **int** columns, **array**<**float**> data) =>

m = matrix.new<**float**>(rows, columns)

for i = 0 to rows <= 0 ? na : rows - 1

for j = 0 to columns <= 0 ? na : columns - 1

matrix.set(m, i, j, array.get(data, i \* columns + j))

m

// Create a 3x3 matrix from an array of values.

var m1 = matrixFromArray(3, 3, array.from(1, 2, 3, 4, 5, 6, 7, 8, 9))

// Display using a label.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, str.tostring(m1))

Create a matrix from an input.text\_area() field

EXAMPLE

//**@version=**6

indicator("`matrix.new<type>()` Example 3")

// Function to create a matrix from a text string.

// Values in a row must be separated by a space. Each line is one row.

matrixFromInputArea(stringOfValues) =>

var rowsArray = str.split(stringOfValues, "\n")

var rows = array.size(rowsArray)

var cols = array.size(str.split(array.get(rowsArray, 0), " "))

var matrix = matrix.new<**float**>(rows, cols, na)

row = 0

for rowString in rowsArray

col = 0

values = str.split(rowString, " ")

for val in values

matrix.set(matrix, row, col, str.tonumber(val))

col += 1

row += 1

**matrix**

stringInput = input.text\_area("1 2 3\n4 5 6\n7 8 9")

var m = matrixFromInputArea(stringInput)

// Display using a label.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, str.tostring(m))

Create matrix from random values

EXAMPLE

//**@version=**6

indicator("`matrix.new<type>()` Example 4")

// Function to create a matrix with random values (0.0 to 1.0).

matrixRandom(**int** rows, **int** columns)=>

result = matrix.new<**float**>(rows, columns)

for i = 0 to rows - 1

for j = 0 to columns - 1

matrix.set(result, i, j, math.random())

result

// Create a 2x3 matrix with random values.

var m = matrixRandom(2, 3)

// Display using a label.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, str.tostring(m))

RETURNS

The ID of the new matrix object.

SEE ALSO

[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.fill](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.fill)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)[array.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new%3Ctype%3E)

**matrix.pinv()**2 overloads

The function returns the [pseudoinverse](https://en.wikipedia.org/wiki/Moore%E2%80%93Penrose_inverse) of a matrix.

SYNTAX & OVERLOADS

[matrix.pinv(id) → matrix<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.pinv-0)

[matrix.pinv(id) → matrix<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.pinv-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.pinv()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x2 matrix.

var m1 = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 1)

matrix.set(m1, 0, 1, 2)

matrix.set(m1, 1, 0, 3)

matrix.set(m1, 1, 1, 4)

// Pseudoinverse of the matrix.

var m2 = matrix.pinv(m1)

// Display matrix elements.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Original Matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Pseudoinverse matrix:")

table.cell(t, 1, 1, str.tostring(m2))

RETURNS

A new matrix containing the pseudoinverse of the id matrix.

REMARKS

The function is calculated using a [Moore–Penrose](https://en.wikipedia.org/wiki/Moore%E2%80%93Penrose_inverse#Definition) inverse formula based on singular-value decomposition of a matrix. For non-singular square matrices this function returns the result of [matrix.inv](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.inv).

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.inv](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.inv)

**matrix.pow()**2 overloads

The function calculates the product of the matrix by itself power times.

SYNTAX & OVERLOADS

[matrix.pow(id, power) → matrix<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.pow-0)

[matrix.pow(id, power) → matrix<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.pow-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

**power (series int)** The number of times the matrix will be multiplied by itself.

EXAMPLE

//**@version=**6

indicator("`matrix.pow()` Example")

// Display using a table.

if barstate.islastconfirmedhistory

// Create a 2x2 matrix.

var m1 = matrix.new<**int**>(2, 2, 2)

// Calculate the power of three of the matrix.

var m2 = matrix.pow(m1, 3)

// Display matrix elements.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Original Matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Matrix³:")

table.cell(t, 1, 1, str.tostring(m2))

RETURNS

The product of the id matrix by itself power times.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.mult](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.mult)

**matrix.rank()**

The function calculates the [rank](https://en.wikipedia.org/wiki/Rank_(linear_algebra)) of the matrix.

SYNTAX

matrix.rank(id) → series int

ARGUMENTS

**id (any matrix type)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.rank()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x2 matrix.

var m1 = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 1)

matrix.set(m1, 0, 1, 2)

matrix.set(m1, 1, 0, 3)

matrix.set(m1, 1, 1, 4)

// Get the rank of the matrix.

r = matrix.rank(m1)

// Display matrix elements.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Matrix elements:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Rank of the matrix:")

table.cell(t, 1, 1, str.tostring(r))

RETURNS

The rank of the id matrix.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[str.tostring](https://www.tradingview.com/pine-script-reference/v6/#fun_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

matrix.remove\_col(id, column) → array<type>

ARGUMENTS

**id (any matrix type)** A matrix object.

**column (series int)** The index of the column to be removed. Optional. The default value is [matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns).

EXAMPLE

//**@version=**6

indicator("matrix\_remove\_col", overlay = true)

// Create a 2x2 matrix with ones.

var matrixOrig = matrix.new<**int**>(2, 2, 1)

// Set values to the 'matrixOrig' matrix.

matrix.set(matrixOrig, 0, 1, 2)

matrix.set(matrixOrig, 1, 0, 3)

matrix.set(matrixOrig, 1, 1, 4)

// Create a copy of the 'matrixOrig' matrix.

matrixCopy = matrix.copy(matrixOrig)

// Remove the first column from the `matrixCopy` matrix.

arr = matrix.remove\_col(matrixCopy, 0)

// Display matrix elements.

if barstate.islastconfirmedhistory

var t = table.new(position.top\_right, 3, 2, color.green)

table.cell(t, 0, 0, "Original Matrix:")

table.cell(t, 0, 1, str.tostring(matrixOrig))

table.cell(t, 1, 0, "Removed Elements:")

table.cell(t, 1, 1, str.tostring(arr))

table.cell(t, 2, 0, "Result Matrix:")

table.cell(t, 2, 1, str.tostring(matrixCopy))

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](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.remove_row) function.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.copy](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.copy)[matrix.remove\_row](https://www.tradingview.com/pine-script-reference/v6/#fun_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

matrix.remove\_row(id, row) → array<type>

ARGUMENTS

**id (any matrix type)** A matrix object.

**row (series int)** The index of the row to be deleted. Optional. The default value is [matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows).

EXAMPLE

//**@version=**6

indicator("matrix\_remove\_row", overlay = true)

// Create a 2x2 "int" matrix containing values `1`.

var matrixOrig = matrix.new<**int**>(2, 2, 1)

// Set values to the 'matrixOrig' matrix.

matrix.set(matrixOrig, 0, 1, 2)

matrix.set(matrixOrig, 1, 0, 3)

matrix.set(matrixOrig, 1, 1, 4)

// Create a copy of the 'matrixOrig' matrix.

matrixCopy = matrix.copy(matrixOrig)

// Remove the first row from the matrix `matrixCopy`.

arr = matrix.remove\_row(matrixCopy, 0)

// Display matrix elements.

if barstate.islastconfirmedhistory

var t = table.new(position.top\_right, 3, 2, color.green)

table.cell(t, 0, 0, "Original Matrix:")

table.cell(t, 0, 1, str.tostring(matrixOrig))

table.cell(t, 1, 0, "Removed Elements:")

table.cell(t, 1, 1, str.tostring(arr))

table.cell(t, 2, 0, "Result Matrix:")

table.cell(t, 2, 1, str.tostring(matrixCopy))

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.copy](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.copy)[matrix.remove\_col](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.remove_col)

**matrix.reshape()**

The function rebuilds the id matrix to rows x cols dimensions.

SYNTAX

matrix.reshape(id, rows, columns) → void

ARGUMENTS

**id (any matrix type)** A matrix object.

**rows (series int)** The number of rows of the reshaped matrix.

**columns (series int)** The number of columns of the reshaped matrix.

EXAMPLE

//**@version=**6

indicator("`matrix.reshape()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x3 matrix.

var m1 = matrix.new<**float**>(2, 3)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 1)

matrix.set(m1, 0, 1, 2)

matrix.set(m1, 0, 2, 3)

matrix.set(m1, 1, 0, 4)

matrix.set(m1, 1, 1, 5)

matrix.set(m1, 1, 2, 6)

// Copy the matrix to a new one.

var m2 = matrix.copy(m1)

// Reshape the copy to a 3x2.

matrix.reshape(m2, 3, 2)

// Display using a table.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Original matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Reshaped matrix:")

table.cell(t, 1, 1, str.tostring(m2))

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.add\_row](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.add_row)[matrix.add\_col](https://www.tradingview.com/pine-script-reference/v6/#fun_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

matrix.reverse(id) → void

ARGUMENTS

**id (any matrix type)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.reverse()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Copy the matrix to a new one.

var m1 = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 1)

matrix.set(m1, 0, 1, 2)

matrix.set(m1, 1, 0, 3)

matrix.set(m1, 1, 1, 4)

// Copy matrix elements to a new matrix.

var m2 = matrix.copy(m1)

// Reverse the `m2` copy of the original matrix.

matrix.reverse(m2)

// Display using a table.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Original matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Reversed matrix:")

table.cell(t, 1, 1, str.tostring(m2))

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)[matrix.reshape](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.reshape)

**matrix.row()**

The function creates a one-dimensional array from the elements of a matrix row.

SYNTAX

matrix.row(id, row) → array<type>

ARGUMENTS

**id (any matrix type)** A matrix object.

**row (series int)** Index of the required row.

EXAMPLE

//**@version=**6

indicator("`matrix.row()` Example", "", true)

// Create a 2x3 "float" matrix from `hlc3` values.

m = matrix.new<**float**>(2, 3, hlc3)

// Return an array with the values of the first row of the matrix.

a = matrix.row(m, 0)

// Plot the first value from the array `a`.

plot(array.get(a, 0))

RETURNS

An array ID containing the row values of the id matrix.

REMARKS

Indexing of rows starts at 0.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[array.get](https://www.tradingview.com/pine-script-reference/v6/#fun_array.get)[matrix.col](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.col)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.rows()**

The function returns the number of rows in the matrix.

SYNTAX

matrix.rows(id) → series int

ARGUMENTS

**id (any matrix type)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.rows()` Example")

// Create a 2x6 matrix with values `0`.

var m = matrix.new<**int**>(2, 6, 0)

// Get the quantity of rows in the matrix.

var x = matrix.rows(m)

// Display using a label.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, "Rows: " + str.tostring(x) + "\n" + str.tostring(m))

RETURNS

The number of rows in the matrix id.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.row](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.row)

**matrix.set()**

The function assigns value to the element at the row and column of the id matrix.

SYNTAX

matrix.set(id, row, column, value) → void

ARGUMENTS

**id (any matrix type)** A matrix object.

**row (series int)** The row index of the element to be modified.

**column (series int)** The column index of the element to be modified.

**value (series <type of the matrix's elements>)** The new value to be set.

EXAMPLE

//**@version=**6

indicator("`matrix.set()` Example")

// Create a 2x3 "int" matrix containing values `4`.

m = matrix.new<**int**>(2, 3, 4)

// Replace the value of element at row 1 and column 2 with value `3`.

matrix.set(m, 0, 1, 3)

// Display using a label.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, str.tostring(m))

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.sort()**

The function rearranges the rows in the id matrix following the sorted order of the values in the column.

SYNTAX

matrix.sort(id, column, order) → void

ARGUMENTS

**id (matrix<int/float/string>)** A matrix object to be sorted.

**column (series int)** Index of the column whose sorted values determine the new order of rows. Optional. The default value is 0.

**order (series sort\_order)** The sort order. Possible values: [order.ascending](https://www.tradingview.com/pine-script-reference/v6/#const_order.ascending) (default), [order.descending](https://www.tradingview.com/pine-script-reference/v6/#const_order.descending).

EXAMPLE

//**@version=**6

indicator("`matrix.sort()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x2 matrix.

var m1 = matrix.new<**float**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 3)

matrix.set(m1, 0, 1, 4)

matrix.set(m1, 1, 0, 1)

matrix.set(m1, 1, 1, 2)

// Copy the matrix to a new one.

var m2 = matrix.copy(m1)

// Sort the rows of `m2` using the default arguments (first column and ascending order).

matrix.sort(m2)

// Display using a table.

if barstate.islastconfirmedhistory

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Original matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Sorted matrix:")

table.cell(t, 1, 1, str.tostring(m2))

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.max](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.max)[matrix.min](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.min)[matrix.avg](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.avg)

**matrix.submatrix()**

The function extracts a submatrix of the id matrix within the specified indices.

SYNTAX

matrix.submatrix(id, from\_row, to\_row, from\_column, to\_column) → matrix<type>

ARGUMENTS

**id (any matrix type)** A matrix object.

**from\_row (series int)** Index of the row from which the extraction will begin (inclusive). Optional. The default value is 0.

**to\_row (series int)** Index of the row where the extraction will end (non inclusive). Optional. The default value is [matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows).

**from\_column (series int)** Index of the column from which the extraction will begin (inclusive). Optional. The default value is 0.

**to\_column (series int)** Index of the column where the extraction will end (non inclusive). Optional. The default value is [matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns).

EXAMPLE

//**@version=**6

indicator("`matrix.submatrix()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x3 matrix matrix with values `0`.

var m1 = matrix.new<**int**>(2, 3, 0)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 1)

matrix.set(m1, 0, 1, 2)

matrix.set(m1, 0, 2, 3)

matrix.set(m1, 1, 0, 4)

matrix.set(m1, 1, 1, 5)

matrix.set(m1, 1, 2, 6)

// Create a 2x2 submatrix of the `m1` matrix.

var m2 = matrix.submatrix(m1, 0, 2, 1, 3)

// Display using a table.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Original Matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Submatrix:")

table.cell(t, 1, 1, str.tostring(m2))

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

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.row](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.row)[matrix.col](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.col)[matrix.reshape](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.reshape)

**matrix.sum()**2 overloads

The function returns a new matrix resulting from the [sum](https://en.wikipedia.org/wiki/Matrix_addition) of two matrices id1 and id2, or of an id1 matrix and an id2 scalar (a numerical value).

SYNTAX & OVERLOADS

[matrix.sum(id1, id2) → matrix<int>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.sum-0)

[matrix.sum(id1, id2) → matrix<float>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.sum-1)

ARGUMENTS

**id1 (matrix<int>)** First matrix object.

**id2 (series int/float/matrix<int>)** Second matrix object, or scalar value.

Sum of two matrices

EXAMPLE

//**@version=**6

indicator("`matrix.sum()` Example 1")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x3 matrix containing values `5`.

var m1 = matrix.new<**float**>(2, 3, 5)

// Create a 2x3 matrix containing values `4`.

var m2 = matrix.new<**float**>(2, 3, 4)

// Create a new matrix that sums matrices `m1` and `m2`.

var m3 = matrix.sum(m1, m2)

// Display using a table.

var t = table.new(position.top\_right, 1, 2, color.green)

table.cell(t, 0, 0, "Sum of two matrices:")

table.cell(t, 0, 1, str.tostring(m3))

Sum of a matrix and scalar

EXAMPLE

//**@version=**6

indicator("`matrix.sum()` Example 2")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x3 matrix with values `4`.

var m1 = matrix.new<**float**>(2, 3, 4)

// Create a new matrix containing the sum of the `m1` matrix with the "int" value `1`.

var m2 = matrix.sum(m1, 1)

// Display using a table.

var t = table.new(position.top\_right, 1, 2, color.green)

table.cell(t, 0, 0, "Sum of a matrix and a scalar:")

table.cell(t, 0, 1, str.tostring(m2))

RETURNS

A new matrix object containing the sum of id2 and id1.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.swap\_columns()**

The function swaps the columns at the index column1 and column2 in the id matrix.

SYNTAX

matrix.swap\_columns(id, column1, column2) → void

ARGUMENTS

**id (any matrix type)** A matrix object.

**column1 (series int)** Index of the first column to be swapped.

**column2 (series int)** Index of the second column to be swapped.

EXAMPLE

//**@version=**6

indicator("`matrix.swap\_columns()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x2 matrix with ‘na’ values.

var m1 = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 1)

matrix.set(m1, 0, 1, 2)

matrix.set(m1, 1, 0, 3)

matrix.set(m1, 1, 1, 4)

// Copy the matrix to a new one.

var m2 = matrix.copy(m1)

// Swap the first and second columns of the matrix copy.

matrix.swap\_columns(m2, 0, 1)

// Display using a table.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Original matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Swapped columns in copy:")

table.cell(t, 1, 1, str.tostring(m2))

REMARKS

Indexing of the rows and columns starts at zero.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.swap\_rows()**

The function swaps the rows at the index row1 and row2 in the id matrix.

SYNTAX

matrix.swap\_rows(id, row1, row2) → void

ARGUMENTS

**id (any matrix type)** A matrix object.

**row1 (series int)** Index of the first row to be swapped.

**row2 (series int)** Index of the second row to be swapped.

EXAMPLE

//**@version=**6

indicator("`matrix.swap\_rows()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 3x2 matrix with ‘na’ values.

var m1 = matrix.new<**int**>(3, 2, na)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 1)

matrix.set(m1, 0, 1, 2)

matrix.set(m1, 1, 0, 3)

matrix.set(m1, 1, 1, 4)

matrix.set(m1, 2, 0, 5)

matrix.set(m1, 2, 1, 6)

// Copy the matrix to a new one.

var m2 = matrix.copy(m1)

// Swap the first and second rows of the matrix copy.

matrix.swap\_rows(m2, 0, 1)

// Display using a table.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Original matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Swapped rows in copy:")

table.cell(t, 1, 1, str.tostring(m2))

REMARKS

Indexing of the rows and columns starts at zero.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.swap\_columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.swap_columns)

**matrix.trace()**2 overloads

The function calculates the [trace](https://en.wikipedia.org/wiki/Trace_(linear_algebra)) of a matrix (the sum of the main diagonal's elements).

SYNTAX & OVERLOADS

[matrix.trace(id) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.trace-0)

[matrix.trace(id) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.trace-1)

ARGUMENTS

**id (matrix<int/float>)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.trace()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x2 matrix.

var m1 = matrix.new<**int**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 1)

matrix.set(m1, 0, 1, 2)

matrix.set(m1, 1, 0, 3)

matrix.set(m1, 1, 1, 4)

// Get the trace of the matrix.

tr = matrix.trace(m1)

// Display matrix elements.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Matrix elements:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Trace of the matrix:")

table.cell(t, 1, 1, str.tostring(tr))

RETURNS

The trace of the id matrix.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.get](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.get)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)

**matrix.transpose()**

The function creates a new, [transposed](https://en.wikipedia.org/wiki/Transpose#Transpose_of_a_matrix) version of the id. This interchanges the row and column index of each element.

SYNTAX

matrix.transpose(id) → matrix<type>

ARGUMENTS

**id (any matrix type)** A matrix object.

EXAMPLE

//**@version=**6

indicator("`matrix.transpose()` Example")

// For efficiency, execute this code only once.

if barstate.islastconfirmedhistory

// Create a 2x2 matrix.

var m1 = matrix.new<**float**>(2, 2, na)

// Fill the matrix with values.

matrix.set(m1, 0, 0, 1)

matrix.set(m1, 0, 1, 2)

matrix.set(m1, 1, 0, 3)

matrix.set(m1, 1, 1, 4)

// Create a transpose of the matrix.

var m2 = matrix.transpose(m1)

// Display using a table.

var t = table.new(position.top\_right, 2, 2, color.green)

table.cell(t, 0, 0, "Original matrix:")

table.cell(t, 0, 1, str.tostring(m1))

table.cell(t, 1, 0, "Transposed matrix:")

table.cell(t, 1, 1, str.tostring(m2))

RETURNS

A new matrix containing the transposed version of the id matrix.

SEE ALSO

[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[matrix.set](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.set)[matrix.columns](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.columns)[matrix.rows](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.rows)[matrix.reshape](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.reshape)[matrix.reverse](https://www.tradingview.com/pine-script-reference/v6/#fun_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.

If an argument of an operator '[]' is a compile time constant value (e.g. 'v[10]', 'close[500]') then there is no need to use 'max\_bars\_back' function for that variable. Pine Script® compiler will use that constant value as history buffer size.

If an argument of an operator '[]' is a value, calculated at runtime (e.g. 'v[i]' where 'i' - is a series variable) then Pine Script® attempts to autodetect the history buffer size at runtime. Sometimes it fails and the script crashes at runtime because it eventually refers to historical values that are out of the buffer. In that case you should use 'max\_bars\_back' to fix that problem manually.

SYNTAX

max\_bars\_back(var, num) → void

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.

**num (const int)** History buffer size which is the number of bars that could be referenced for variable 'var'.

EXAMPLE

//**@version=**6

indicator("max\_bars\_back")

close\_() => close

depth() => 400

d = depth()

v = close\_()

max\_bars\_back(v, 500)

out = if bar\_index > 0

v[d]

else

v

plot(out)

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)’).

If the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) or [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) 'max\_bars\_back' parameter is used, all variables in the indicator are affected. This may result in excessive memory usage and cause runtime problems. When possible (i.e. when the cause is a variable rather than a function), please use the [max\_bars\_back](https://www.tradingview.com/pine-script-reference/v6/#fun_max_bars_back) function instead.

SEE ALSO

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)

**minute()**2 overloads

SYNTAX & OVERLOADS

[minute(time) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_minute-0)

[minute(time, timezone) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_minute-1)

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

[minute](https://www.tradingview.com/pine-script-reference/v6/#var_minute)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[year](https://www.tradingview.com/pine-script-reference/v6/#fun_year)[month](https://www.tradingview.com/pine-script-reference/v6/#fun_month)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#fun_hour)[second](https://www.tradingview.com/pine-script-reference/v6/#fun_second)

**month()**2 overloads

SYNTAX & OVERLOADS

[month(time) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_month-0)

[month(time, timezone) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_month-1)

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.

Note that this function 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 UTC-4) this value can be lower by 1 than the month of the trading day.

SEE ALSO

[month](https://www.tradingview.com/pine-script-reference/v6/#var_month)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[year](https://www.tradingview.com/pine-script-reference/v6/#fun_year)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#fun_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#fun_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#fun_second)

**na()**2 overloads

Tests if x is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

SYNTAX & OVERLOADS

[na(x) → simple bool](https://www.tradingview.com/pine-script-reference/v6/#fun_na-0)

[na(x) → series bool](https://www.tradingview.com/pine-script-reference/v6/#fun_na-1)

ARGUMENTS

**x (simple int/float)** Value to be tested.

EXAMPLE

//**@version=**6

indicator("na")

// Use the `na()` function to test for `na`.

plot(na(close[1]) ? close : close[1])

// ALTERNATIVE

// `nz()` also tests `close[1]` for `na`. It returns `close[1]` if it is not `na`, and `close` if it is.

plot(nz(close[1], close))

RETURNS

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) if x is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

SEE ALSO

[na](https://www.tradingview.com/pine-script-reference/v6/#var_na)[fixnan](https://www.tradingview.com/pine-script-reference/v6/#fun_fixnan)[nz](https://www.tradingview.com/pine-script-reference/v6/#fun_nz)

**nz()**12 overloads

Replaces NaN values with zeros (or given value) in a series.

SYNTAX & OVERLOADS

[nz(source) → simple color](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-0)

[nz(source) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-1)

[nz(source) → series color](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-2)

[nz(source) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-3)

[nz(source) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-4)

[nz(source) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-5)

[nz(source, replacement) → simple color](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-6)

[nz(source, replacement) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-7)

[nz(source, replacement) → series color](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-8)

[nz(source, replacement) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-9)

[nz(source, replacement) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-10)

[nz(source, replacement) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_nz-11)

ARGUMENTS

**source (simple color)** Series of values to process.

EXAMPLE

//**@version=**6

indicator("nz", overlay=true)

plot(nz(ta.sma(close, 100)))

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

[na](https://www.tradingview.com/pine-script-reference/v6/#var_na)[na](https://www.tradingview.com/pine-script-reference/v6/#fun_na)[fixnan](https://www.tradingview.com/pine-script-reference/v6/#fun_fixnan)

**plot()**

Plots a series of data on the chart.

SYNTAX

plot(series, title, color, linewidth, style, trackprice, histbase, offset, join, editable, show\_last, display, format, precision, force\_overlay) → plot

ARGUMENTS

**series (series int/float)** Series of data to be plotted. Required argument.

**title (const string)** Title of the plot.

**color (series color)** Color of the plot. You can use constants like 'color=color.red' or 'color=#ff001a' as well as complex expressions like 'color = close >= open ? color.green : color.red'. Optional argument.

**linewidth (input int)** Width of the plotted line. Default value is 1. Not applicable to every style.

**style (input plot\_style)** Type of plot. Possible values are: [plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line), [plot.style\_stepline](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline), [plot.style\_stepline\_diamond](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_stepline_diamond), [plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram), [plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross), [plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area), [plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns), [plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_circles), [plot.style\_linebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_linebr), [plot.style\_areabr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_areabr), [plot.style\_steplinebr](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_steplinebr). Default value is [plot.style\_line](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_line).

**trackprice (input bool)** If true then a horizontal price line will be shown at the level of the last indicator value. Default is false.

**histbase (input int/float)** The price value used as the reference level when rendering plot with [plot.style\_histogram](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_histogram), [plot.style\_columns](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_columns) or [plot.style\_area](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_area) style. Default is 0.0.

**offset (simple int)** Shifts the plot to the left or to the right on the given number of bars. Default is 0.

**join (input bool)** If true then plot points will be joined with line, applicable only to [plot.style\_cross](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_cross) and [plot.style\_circles](https://www.tradingview.com/pine-script-reference/v6/#const_plot.style_circles) styles. Default is false.

**editable (input bool)** If true then plot style will be editable in Format dialog. Default is true.

**show\_last (input int)** Optional. The number of bars, counting backwards from the most recent bar, on which the function can draw.

**display (input plot\_display)** Controls where the plot's information is displayed. Display options support addition and subtraction, meaning that using display.all - display.status\_line will display the plot's information everywhere except in the script's status line. display.price\_scale + display.status\_line will display the plot only in the price scale and status line. When display arguments such as display.price\_scale have user-controlled chart settings equivalents, the relevant plot information will only appear when all settings allow for it. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.pane](https://www.tradingview.com/pine-script-reference/v6/#const_display.pane), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.price\_scale](https://www.tradingview.com/pine-script-reference/v6/#const_display.price_scale), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

**format (input string)** Determines whether the script formats the plot's values as prices, percentages, or volume values. The argument passed to this parameter supersedes the format parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator), and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. Optional. The default is the format value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function. Possible values: [format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price), [format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent), [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume).

**precision (input int)** The number of digits after the decimal point the plot's values show on the chart pane's y-axis, the script's status line, and the Data Window. Accepts a non-negative integer less than or equal to 16. The argument passed to this parameter supersedes the precision parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. When the function's format parameter uses [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume), the precision parameter will not affect the result, as the decimal precision rules defined by [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume) supersede other precision settings. Optional. The default is the precision value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function.

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the plotted results will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("plot")

plot(high+low, title='Title', color=color.new(#00ffaa, 70), linewidth=2, style=plot.style\_area, offset=15, trackprice=true)

// You may fill the background between any two plots with a fill() function:

p1 = plot(open)

p2 = plot(close)

fill(p1, p2, color=color.new(color.green, 90))

RETURNS

A plot object, that can be used in [fill](https://www.tradingview.com/pine-script-reference/v6/#fun_fill)

SEE ALSO

[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[plotarrow](https://www.tradingview.com/pine-script-reference/v6/#fun_plotarrow)[barcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_barcolor)[bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_bgcolor)[fill](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#var_na) then no arrow is drawn. Arrows has different height, the more absolute indicator value the longer arrow is drawn.

SYNTAX

plotarrow(series, title, colorup, colordown, offset, minheight, maxheight, editable, show\_last, display, format, precision, force\_overlay) → void

ARGUMENTS

**series (series int/float)** Series of data to be plotted as arrows. Required argument.

**title (const string)** Title of the plot.

**colorup (series color)** Color of the up arrows. Optional argument.

**colordown (series color)** Color of the down arrows. Optional argument.

**offset (simple int)** Shifts arrows to the left or to the right on the given number of bars. Default is 0.

**minheight (input int)** Minimal possible arrow height in pixels. Default is 5.

**maxheight (input int)** Maximum possible arrow height in pixels. Default is 100.

**editable (input bool)** If true then plotarrow style will be editable in Format dialog. Default is true.

**show\_last (input int)** Optional. The number of bars, counting backwards from the most recent bar, on which the function can draw.

**display (input plot\_display)** Controls where the plot's information is displayed. Display options support addition and subtraction, meaning that using display.all - display.status\_line will display the plot's information everywhere except in the script's status line. display.price\_scale + display.status\_line will display the plot only in the price scale and status line. When display arguments such as display.price\_scale have user-controlled chart settings equivalents, the relevant plot information will only appear when all settings allow for it. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.pane](https://www.tradingview.com/pine-script-reference/v6/#const_display.pane), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.price\_scale](https://www.tradingview.com/pine-script-reference/v6/#const_display.price_scale), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

**format (input string)** Determines whether the script formats the plot's values as prices, percentages, or volume values. The argument passed to this parameter supersedes the format parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator), and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. Optional. The default is the format value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function. Possible values: [format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price), [format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent), [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume).

**precision (input int)** The number of digits after the decimal point the plot's values show on the chart pane's y-axis, the script's status line, and the Data Window. Accepts a non-negative integer less than or equal to 16. The argument passed to this parameter supersedes the precision parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. When the function's format parameter uses [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume), the precision parameter will not affect the result, as the decimal precision rules defined by [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume) supersede other precision settings. Optional. The default is the precision value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function.

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the plotted results will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("plotarrow example", overlay=true)

codiff = close - open

plotarrow(codiff, colorup=color.new(color.teal,40), colordown=color.new(color.orange, 40))

REMARKS

Use [plotarrow](https://www.tradingview.com/pine-script-reference/v6/#fun_plotarrow) function in conjunction with 'overlay=true' [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) parameter!

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[barcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_barcolor)[bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_bgcolor)

**plotbar()**

Plots ohlc bars on the chart.

SYNTAX

plotbar(open, high, low, close, title, color, editable, show\_last, display, format, precision, force\_overlay) → void

ARGUMENTS

**open (series int/float)** Open series of data to be used as open values of bars. Required argument.

**high (series int/float)** High series of data to be used as high values of bars. Required argument.

**low (series int/float)** Low series of data to be used as low values of bars. Required argument.

**close (series int/float)** Close series of data to be used as close values of bars. Required argument.

**title (const string)** Title of the plotbar. Optional argument.

**color (series color)** Color of the ohlc bars. You can use constants like 'color=color.red' or 'color=#ff001a' as well as complex expressions like 'color = close >= open ? color.green : color.red'. Optional argument.

**editable (const bool)** If true then plotbar style will be editable in Format dialog. Default is true.

**show\_last (input int)** Optional. The number of bars, counting backwards from the most recent bar, on which the function can draw.

**display (input plot\_display)** Controls where the plot's information is displayed. Display options support addition and subtraction, meaning that using display.all - display.status\_line will display the plot's information everywhere except in the script's status line. display.price\_scale + display.status\_line will display the plot only in the price scale and status line. When display arguments such as display.price\_scale have user-controlled chart settings equivalents, the relevant plot information will only appear when all settings allow for it. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.pane](https://www.tradingview.com/pine-script-reference/v6/#const_display.pane), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.price\_scale](https://www.tradingview.com/pine-script-reference/v6/#const_display.price_scale), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

**format (input string)** Determines whether the script formats the plot's values as prices, percentages, or volume values. The argument passed to this parameter supersedes the format parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator), and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. Optional. The default is the format value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function. Possible values: [format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price), [format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent), [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume).

**precision (input int)** The number of digits after the decimal point the plot's values show on the chart pane's y-axis, the script's status line, and the Data Window. Accepts a non-negative integer less than or equal to 16. The argument passed to this parameter supersedes the precision parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. When the function's format parameter uses [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume), the precision parameter will not affect the result, as the decimal precision rules defined by [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume) supersede other precision settings. Optional. The default is the precision value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function.

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the plotted results will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("plotbar example", overlay=true)

plotbar(open, high, low, close, title='Title', color = open < close ? color.green : color.red)

REMARKS

Even if one value of open, high, low or close equal NaN then bar no draw.

The maximal value of open, high, low or close will be set as 'high', and the minimal value will be set as 'low'.

SEE ALSO

[plotcandle](https://www.tradingview.com/pine-script-reference/v6/#fun_plotcandle)

**plotcandle()**

Plots candles on the chart.

SYNTAX

plotcandle(open, high, low, close, title, color, wickcolor, editable, show\_last, bordercolor, display, format, precision, force\_overlay) → void

ARGUMENTS

**open (series int/float)** Open series of data to be used as open values of candles. Required argument.

**high (series int/float)** High series of data to be used as high values of candles. Required argument.

**low (series int/float)** Low series of data to be used as low values of candles. Required argument.

**close (series int/float)** Close series of data to be used as close values of candles. Required argument.

**title (const string)** Title of the plotcandles. Optional argument.

**color (series color)** Color of the candles. You can use constants like 'color=color.red' or 'color=#ff001a' as well as complex expressions like 'color = close >= open ? color.green : color.red'. Optional argument.

**wickcolor (series color)** The color of the wick of candles. An optional argument.

**editable (const bool)** If true then plotcandle style will be editable in Format dialog. Default is true.

**show\_last (input int)** Optional. The number of bars, counting backwards from the most recent bar, on which the function can draw.

**bordercolor (series color)** The border color of candles. An optional argument.

**display (input plot\_display)** Controls where the plot's information is displayed. Display options support addition and subtraction, meaning that using display.all - display.status\_line will display the plot's information everywhere except in the script's status line. display.price\_scale + display.status\_line will display the plot only in the price scale and status line. When display arguments such as display.price\_scale have user-controlled chart settings equivalents, the relevant plot information will only appear when all settings allow for it. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.pane](https://www.tradingview.com/pine-script-reference/v6/#const_display.pane), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.price\_scale](https://www.tradingview.com/pine-script-reference/v6/#const_display.price_scale), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

**format (input string)** Determines whether the script formats the plot's values as prices, percentages, or volume values. The argument passed to this parameter supersedes the format parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator), and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. Optional. The default is the format value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function. Possible values: [format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price), [format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent), [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume).

**precision (input int)** The number of digits after the decimal point the plot's values show on the chart pane's y-axis, the script's status line, and the Data Window. Accepts a non-negative integer less than or equal to 16. The argument passed to this parameter supersedes the precision parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. When the function's format parameter uses [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume), the precision parameter will not affect the result, as the decimal precision rules defined by [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume) supersede other precision settings. Optional. The default is the precision value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function.

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the plotted results will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("plotcandle example", overlay=true)

plotcandle(open, high, low, close, title='Title', color = open < close ? color.green : color.red, wickcolor=color.black)

REMARKS

Even if one value of open, high, low or close equal NaN then bar no draw.

The maximal value of open, high, low or close will be set as 'high', and the minimal value will be set as 'low'.

SEE ALSO

[plotbar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotbar)

**plotchar()**

Plots visual shapes using any given one Unicode character on the chart.

SYNTAX

plotchar(series, title, char, location, color, offset, text, textcolor, editable, size, show\_last, display, format, precision, force\_overlay) → void

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](https://www.tradingview.com/pine-script-reference/v6/#const_location.absolute). Required argument.

**title (const string)** Title of the plot.

**char (input string)** Character to use as a visual shape.

**location (input string)** Location of shapes on the chart. Possible values are: [location.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_location.abovebar), [location.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_location.belowbar), [location.top](https://www.tradingview.com/pine-script-reference/v6/#const_location.top), [location.bottom](https://www.tradingview.com/pine-script-reference/v6/#const_location.bottom), [location.absolute](https://www.tradingview.com/pine-script-reference/v6/#const_location.absolute). Default value is [location.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_location.abovebar).

**color (series color)** Color of the shapes. You can use constants like 'color=color.red' or 'color=#ff001a' as well as complex expressions like 'color = close >= open ? color.green : color.red'. Optional argument.

**offset (simple int)** Shifts shapes to the left or to the right on the given number of bars. Default is 0.

**text (const string)** Text to display with the shape. You can use multiline text, to separate lines use '\n' escape sequence. Example: 'line one\nline two'.

**textcolor (series color)** Color of the text. You can use constants like 'textcolor=color.red' or 'textcolor=#ff001a' as well as complex expressions like 'textcolor = close >= open ? color.green : color.red'. Optional argument.

**editable (const bool)** If true then plotchar style will be editable in Format dialog. Default is true.

**size (const string)** Size of characters on the chart. Possible values are: [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto), [size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny), [size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small), [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal), [size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large), [size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge). Default is [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto).

**show\_last (input int)** Optional. The number of bars, counting backwards from the most recent bar, on which the function can draw.

**display (input plot\_display)** Controls where the plot's information is displayed. Display options support addition and subtraction, meaning that using display.all - display.status\_line will display the plot's information everywhere except in the script's status line. display.price\_scale + display.status\_line will display the plot only in the price scale and status line. When display arguments such as display.price\_scale have user-controlled chart settings equivalents, the relevant plot information will only appear when all settings allow for it. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.pane](https://www.tradingview.com/pine-script-reference/v6/#const_display.pane), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.price\_scale](https://www.tradingview.com/pine-script-reference/v6/#const_display.price_scale), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

**format (input string)** Determines whether the script formats the plot's values as prices, percentages, or volume values. The argument passed to this parameter supersedes the format parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator), and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. Optional. The default is the format value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function. Possible values: [format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price), [format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent), [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume).

**precision (input int)** The number of digits after the decimal point the plot's values show on the chart pane's y-axis, the script's status line, and the Data Window. Accepts a non-negative integer less than or equal to 16. The argument passed to this parameter supersedes the precision parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. When the function's format parameter uses [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume), the precision parameter will not affect the result, as the decimal precision rules defined by [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume) supersede other precision settings. Optional. The default is the precision value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function.

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the plotted results will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("plotchar example", overlay=true)

data = close >= open

plotchar(data, char='❄')

REMARKS

Use [plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar) function in conjunction with 'overlay=true' [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) parameter!

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape)[plotarrow](https://www.tradingview.com/pine-script-reference/v6/#fun_plotarrow)[barcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_barcolor)[bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_bgcolor)

**plotshape()**

Plots visual shapes on the chart.

SYNTAX

plotshape(series, title, style, location, color, offset, text, textcolor, editable, size, show\_last, display, format, precision, force\_overlay) → void

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](https://www.tradingview.com/pine-script-reference/v6/#const_location.absolute). Required argument.

**title (const string)** Title of the plot.

**style (input string)** Type of plot. Possible values are: [shape.xcross](https://www.tradingview.com/pine-script-reference/v6/#const_shape.xcross), [shape.cross](https://www.tradingview.com/pine-script-reference/v6/#const_shape.cross), [shape.triangleup](https://www.tradingview.com/pine-script-reference/v6/#const_shape.triangleup), [shape.triangledown](https://www.tradingview.com/pine-script-reference/v6/#const_shape.triangledown), [shape.flag](https://www.tradingview.com/pine-script-reference/v6/#const_shape.flag), [shape.circle](https://www.tradingview.com/pine-script-reference/v6/#const_shape.circle), [shape.arrowup](https://www.tradingview.com/pine-script-reference/v6/#const_shape.arrowup), [shape.arrowdown](https://www.tradingview.com/pine-script-reference/v6/#const_shape.arrowdown), [shape.labelup](https://www.tradingview.com/pine-script-reference/v6/#const_shape.labelup), [shape.labeldown](https://www.tradingview.com/pine-script-reference/v6/#const_shape.labeldown), [shape.square](https://www.tradingview.com/pine-script-reference/v6/#const_shape.square), [shape.diamond](https://www.tradingview.com/pine-script-reference/v6/#const_shape.diamond). Default value is [shape.xcross](https://www.tradingview.com/pine-script-reference/v6/#const_shape.xcross).

**location (input string)** Location of shapes on the chart. Possible values are: [location.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_location.abovebar), [location.belowbar](https://www.tradingview.com/pine-script-reference/v6/#const_location.belowbar), [location.top](https://www.tradingview.com/pine-script-reference/v6/#const_location.top), [location.bottom](https://www.tradingview.com/pine-script-reference/v6/#const_location.bottom), [location.absolute](https://www.tradingview.com/pine-script-reference/v6/#const_location.absolute). Default value is [location.abovebar](https://www.tradingview.com/pine-script-reference/v6/#const_location.abovebar).

**color (series color)** Color of the shapes. You can use constants like 'color=color.red' or 'color=#ff001a' as well as complex expressions like 'color = close >= open ? color.green : color.red'. Optional argument.

**offset (simple int)** Shifts shapes to the left or to the right on the given number of bars. Default is 0.

**text (const string)** Text to display with the shape. You can use multiline text, to separate lines use '\n' escape sequence. Example: 'line one\nline two'.

**textcolor (series color)** Color of the text. You can use constants like 'textcolor=color.red' or 'textcolor=#ff001a' as well as complex expressions like 'textcolor = close >= open ? color.green : color.red'. Optional argument.

**editable (const bool)** If true then plotshape style will be editable in Format dialog. Default is true.

**size (const string)** Size of shapes on the chart. Possible values are: [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto), [size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny), [size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small), [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal), [size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large), [size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge). Default is [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto).

**show\_last (input int)** Optional. The number of bars, counting backwards from the most recent bar, on which the function can draw.

**display (input plot\_display)** Controls where the plot's information is displayed. Display options support addition and subtraction, meaning that using display.all - display.status\_line will display the plot's information everywhere except in the script's status line. display.price\_scale + display.status\_line will display the plot only in the price scale and status line. When display arguments such as display.price\_scale have user-controlled chart settings equivalents, the relevant plot information will only appear when all settings allow for it. Possible values: [display.none](https://www.tradingview.com/pine-script-reference/v6/#const_display.none), [display.pane](https://www.tradingview.com/pine-script-reference/v6/#const_display.pane), [display.data\_window](https://www.tradingview.com/pine-script-reference/v6/#const_display.data_window), [display.price\_scale](https://www.tradingview.com/pine-script-reference/v6/#const_display.price_scale), [display.status\_line](https://www.tradingview.com/pine-script-reference/v6/#const_display.status_line), [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all). Optional. The default is [display.all](https://www.tradingview.com/pine-script-reference/v6/#const_display.all).

**format (input string)** Determines whether the script formats the plot's values as prices, percentages, or volume values. The argument passed to this parameter supersedes the format parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator), and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. Optional. The default is the format value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function. Possible values: [format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price), [format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent), [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume).

**precision (input int)** The number of digits after the decimal point the plot's values show on the chart pane's y-axis, the script's status line, and the Data Window. Accepts a non-negative integer less than or equal to 16. The argument passed to this parameter supersedes the precision parameter of the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) and [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) functions. When the function's format parameter uses [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume), the precision parameter will not affect the result, as the decimal precision rules defined by [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume) supersede other precision settings. Optional. The default is the precision value used by the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)/[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function.

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the plotted results will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("plotshape example 1", overlay=true)

data = close >= open

plotshape(data, style=shape.xcross)

REMARKS

Use [plotshape](https://www.tradingview.com/pine-script-reference/v6/#fun_plotshape) function in conjunction with 'overlay=true' [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) parameter!

SEE ALSO

[plot](https://www.tradingview.com/pine-script-reference/v6/#fun_plot)[plotchar](https://www.tradingview.com/pine-script-reference/v6/#fun_plotchar)[plotarrow](https://www.tradingview.com/pine-script-reference/v6/#fun_plotarrow)[barcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_barcolor)[bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_bgcolor)

**polyline.delete()**

Deletes the specified [polyline](https://www.tradingview.com/pine-script-reference/v6/#type_polyline) object. It has no effect if the id doesn't exist.

SYNTAX

polyline.delete(id) → void

ARGUMENTS

**id (series polyline)** The polyline ID to delete.

**polyline.new()**

Creates a new [polyline](https://www.tradingview.com/pine-script-reference/v6/#type_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

polyline.new(points, curved, closed, xloc, line\_color, fill\_color, line\_style, line\_width, force\_overlay) → series polyline

ARGUMENTS

**points (array<chart.point>)** An array of [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) objects for the drawing to sequentially connect.

**curved (series bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the drawing will connect all points from the points array using curved line segments. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**closed (series bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the drawing will also connect the first point to the last point from the points array, resulting in a closed polyline. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**xloc (series string)** Determines the field of the [chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_chart.point) objects in the points array that the polyline will use for its x-coordinates. If [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index), the polyline will use the index field from each point. If [xloc.bar\_time](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_time), it will use the time field. Optional. The default is [xloc.bar\_index](https://www.tradingview.com/pine-script-reference/v6/#const_xloc.bar_index).

**line\_color (series color)** The color of the line segments. Optional. The default is [color.blue](https://www.tradingview.com/pine-script-reference/v6/#const_color.blue).

**fill\_color (series color)** The fill color of the polyline. Optional. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**line\_style (series string)** The style of the polyline. Possible values: [line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid), [line.style\_dotted](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dotted), [line.style\_dashed](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_dashed), [line.style\_arrow\_left](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_left), [line.style\_arrow\_right](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_right), [line.style\_arrow\_both](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_arrow_both). Optional. The default is [line.style\_solid](https://www.tradingview.com/pine-script-reference/v6/#const_line.style_solid).

**line\_width (series int)** The width of the line segments, expressed in pixels. Optional. The default is 1.

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the drawing will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("Polylines example", overlay = true)

//**@variable** If `true`, connects all points in the polyline with curved line segments.

**bool** curvedInput = input.bool(false, "Curve Polyline")

//**@variable** If `true`, connects the first point in the polyline to the last point.

**bool** closedInput = input.bool(true, "Close Polyline")

//**@variable** The color of the space filled by the polyline.

**color** fillcolor = input.color(color.new(color.blue, 90), "Fill Color")

// Time and price inputs for the polyline's points.

p1x = input.time(0, "p1", confirm = true, inline = "p1")

p1y = input.price(0, " ", confirm = true, inline = "p1")

p2x = input.time(0, "p2", confirm = true, inline = "p2")

p2y = input.price(0, " ", confirm = true, inline = "p2")

p3x = input.time(0, "p3", confirm = true, inline = "p3")

p3y = input.price(0, " ", confirm = true, inline = "p3")

p4x = input.time(0, "p4", confirm = true, inline = "p4")

p4y = input.price(0, " ", confirm = true, inline = "p4")

p5x = input.time(0, "p5", confirm = true, inline = "p5")

p5y = input.price(0, " ", confirm = true, inline = "p5")

if barstate.islastconfirmedhistory

//**@variable** An array of `chart.point` objects for the new polyline.

var points = array.new<**chart.point**>()

// Push new `chart.point` instances into the `points` array.

points.push(chart.point.from\_time(p1x, p1y))

points.push(chart.point.from\_time(p2x, p2y))

points.push(chart.point.from\_time(p3x, p3y))

points.push(chart.point.from\_time(p4x, p4y))

points.push(chart.point.from\_time(p5x, p5y))

// Add labels for each `chart.point` in `points`.

l1p1 = label.new(points.get(0), text = "p1", xloc = xloc.bar\_time, color = na)

l1p2 = label.new(points.get(1), text = "p2", xloc = xloc.bar\_time, color = na)

l2p1 = label.new(points.get(2), text = "p3", xloc = xloc.bar\_time, color = na)

l2p2 = label.new(points.get(3), text = "p4", xloc = xloc.bar\_time, color = na)

// Create a new polyline that connects each `chart.point` in the `points` array, starting from the first.

polyline.new(points, curved = curvedInput, closed = closedInput, fill\_color = fillcolor, xloc = xloc.bar\_time)

RETURNS

The ID of a new polyline object that a script can use in other polyline.\*() functions.

SEE ALSO

[chart.point.new](https://www.tradingview.com/pine-script-reference/v6/#fun_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

request.currency\_rate(from, to, ignore\_invalid\_currency) → series float

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](https://en.wikipedia.org/wiki/ISO_4217#Active_codes) (e.g. "USD"), or one of the built-in variables that return currency codes, like [syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency) or [currency.USD](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USD).

**to (series string)** The currency in which the value is to be converted. Possible values: a three-letter string with the [currency code in the ISO 4217 format](https://en.wikipedia.org/wiki/ISO_4217#Active_codes) (e.g. "USD"), or one of the built-in variables that return currency codes, like [syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency) or [currency.USD](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USD).

**ignore\_invalid\_currency (series bool)** Determines the behavior of the function if a conversion rate between the two currencies cannot be calculated: if [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), the script will halt and return a runtime error; if [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the function will return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) and execution will continue. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("Close in British Pounds")

rate = request.currency\_rate(syminfo.currency, "GBP")

plot(close \* rate)

REMARKS

If from and to arguments are equal, function returns 1. Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

**request.dividends()**

Requests dividends data for the specified symbol.

SYNTAX

request.dividends(ticker, field, gaps, lookahead, ignore\_invalid\_symbol, currency) → series float

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](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker) will cause an error. Use [syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid) instead.

**field (series string)** Input string. Possible values include: [dividends.net](https://www.tradingview.com/pine-script-reference/v6/#const_dividends.net), [dividends.gross](https://www.tradingview.com/pine-script-reference/v6/#const_dividends.gross). Default value is [dividends.gross](https://www.tradingview.com/pine-script-reference/v6/#const_dividends.gross).

**gaps (simple barmerge\_gaps)** Merge strategy for the requested data (requested data automatically merges with the main series OHLC data). Possible values: [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on), [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off). [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on) - requested data is merged with possible gaps ([na](https://www.tradingview.com/pine-script-reference/v6/#var_na) values). [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off) - requested data is merged continuously without gaps, all the gaps are filled with the previous nearest existing values. Default value is [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off).

**lookahead (simple barmerge\_lookahead)** Merge strategy for the requested data position. Possible values: [barmerge.lookahead\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_on), [barmerge.lookahead\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_off). Default value is [barmerge.lookahead\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_off) starting from version 3. Note that behavour is the same on real-time, and differs only on history.

**ignore\_invalid\_symbol (input bool)** An optional parameter. Determines the behavior of the function if the specified symbol is not found: if false, the script will halt and return a runtime error; if true, the function will return na and execution will continue. The default value is false.

**currency (series string)** Currency into which the symbol's currency-related dividends values (e.g. [dividends.gross](https://www.tradingview.com/pine-script-reference/v6/#const_dividends.gross)) are to be converted. The conversion rate depends on the previous daily value of a corresponding currency pair from the most popular exchange. A spread symbol is used if no exchange provides the rate directly. Possible values: a "string" representing a valid currency code (e.g., "USD" or "USDT") or a constant from the currency.\* namespace (e.g., [currency.USD](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USD) or [currency.USDT](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USDT)). The default is [syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency).

EXAMPLE

//**@version=**6

indicator("request.dividends")

s1 = request.dividends("NASDAQ:BELFA")

plot(s1)

s2 = request.dividends("NASDAQ:BELFA", dividends.net, gaps=barmerge.gaps\_on, lookahead=barmerge.lookahead\_on)

plot(s2)

RETURNS

Requested series, or n/a if there is no dividends data for the specified symbol.

SEE ALSO

[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings)[request.splits](https://www.tradingview.com/pine-script-reference/v6/#fun_request.splits)[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)

**request.earnings()**

Requests earnings data for the specified symbol.

SYNTAX

request.earnings(ticker, field, gaps, lookahead, ignore\_invalid\_symbol, currency) → series float

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](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker) will cause an error. Use [syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid) instead.

**field (series string)** Input string. Possible values include: [earnings.actual](https://www.tradingview.com/pine-script-reference/v6/#const_earnings.actual), [earnings.estimate](https://www.tradingview.com/pine-script-reference/v6/#const_earnings.estimate), [earnings.standardized](https://www.tradingview.com/pine-script-reference/v6/#const_earnings.standardized). Default value is [earnings.actual](https://www.tradingview.com/pine-script-reference/v6/#const_earnings.actual).

**gaps (simple barmerge\_gaps)** Merge strategy for the requested data (requested data automatically merges with the main series OHLC data). Possible values: [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on), [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off). [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on) - requested data is merged with possible gaps ([na](https://www.tradingview.com/pine-script-reference/v6/#var_na) values). [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off) - requested data is merged continuously without gaps, all the gaps are filled with the previous nearest existing values. Default value is [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off).

**lookahead (simple barmerge\_lookahead)** Merge strategy for the requested data position. Possible values: [barmerge.lookahead\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_on), [barmerge.lookahead\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_off). Default value is [barmerge.lookahead\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_off) starting from version 3. Note that behavour is the same on real-time, and differs only on history.

**ignore\_invalid\_symbol (input bool)** An optional parameter. Determines the behavior of the function if the specified symbol is not found: if false, the script will halt and return a runtime error; if true, the function will return na and execution will continue. The default value is false.

**currency (series string)** Currency into which the symbol's currency-related earnings values (e.g. [earnings.actual](https://www.tradingview.com/pine-script-reference/v6/#const_earnings.actual)) are to be converted. The conversion rate depends on the previous daily value of a corresponding currency pair from the most popular exchange. A spread symbol is used if no exchange provides the rate directly. Possible values: a "string" representing a valid currency code (e.g., "USD" or "USDT") or a constant from the currency.\* namespace (e.g., [currency.USD](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USD) or [currency.USDT](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USDT)). The default is [syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency).

EXAMPLE

//**@version=**6

indicator("request.earnings")

s1 = request.earnings("NASDAQ:BELFA")

plot(s1)

s2 = request.earnings("NASDAQ:BELFA", earnings.actual, gaps=barmerge.gaps\_on, lookahead=barmerge.lookahead\_on)

plot(s2)

RETURNS

Requested series, or n/a if there is no earnings data for the specified symbol.

SEE ALSO

[request.dividends](https://www.tradingview.com/pine-script-reference/v6/#fun_request.dividends)[request.splits](https://www.tradingview.com/pine-script-reference/v6/#fun_request.splits)[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_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

request.economic(country\_code, field, gaps, ignore\_invalid\_symbol) → series float

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](https://www.tradingview.com/chart/?solution=43000665359) lists the countries and their codes. The countries for which information is available vary with metrics. The [Help Center article for each metric](https://www.tradingview.com/support/folders/43000581956-list-of-available-economic-indicators/) lists the countries for which the metric is available.

**field (series string)** The code of the requested economic metric (e.g., "GDP"). The [Help Center article](https://www.tradingview.com/chart/?solution=43000665359) lists the metrics and their codes.

**gaps (simple barmerge\_gaps)** Specifies how the returned values are merged on chart bars. Possible values: [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off), [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on). With [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on), a value only appears on the current chart bar when it first becomes available from the function's context, otherwise [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) is returned (thus a "gap" occurs). With [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off), what would otherwise be gaps are filled with the latest known value returned, avoiding [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) values. Optional. The default is [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off).

**ignore\_invalid\_symbol (input bool)** Determines the behavior of the function if the specified symbol is not found: if [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), the script will halt and return a runtime error; if [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the function will return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) and execution will continue. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("US GDP")

e = request.economic("US", "GDP")

plot(e)

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

[request.financial](https://www.tradingview.com/pine-script-reference/v6/#fun_request.financial)

**request.financial()**

Requests financial series for symbol.

SYNTAX

request.financial(symbol, financial\_id, period, gaps, ignore\_invalid\_symbol, currency) → series float

ARGUMENTS

**symbol (series string)** Symbol. Note that the symbol should be passed with a prefix. For example: "NASDAQ:AAPL" instead of "AAPL".

**financial\_id (series string)** Financial identifier. You can find the list of available ids via our [Help Center](https://www.tradingview.com/?solution=43000564727).

**period (series string)** Reporting period. Possible values are "TTM", "FY", "FQ", "FH", "D".

**gaps (simple barmerge\_gaps)** Merge strategy for the requested data (requested data automatically merges with the main series: OHLC data). Possible values include: [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on), [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off). [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on) - requested data is merged with possible gaps ([na](https://www.tradingview.com/pine-script-reference/v6/#var_na) values). [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off) - requested data is merged continuously without gaps, all the gaps are filled with the previous, nearest existing values. Default value is [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off).

**ignore\_invalid\_symbol (input bool)** An optional parameter. Determines the behavior of the function if the specified symbol is not found: if false, the script will halt and return a runtime error; if true, the function will return na and execution will continue. The default value is false.

**currency (series string)** Optional. Currency into which the symbol's financial metrics (e.g. Net Income) are to be converted. The conversion rate depends on the previous daily value of a corresponding currency pair from the most popular exchange. A spread symbol is used if no exchange provides the rate directly. Possible values: a "string" representing a valid currency code (e.g., "USD" or "USDT") or a constant from the currency.\* namespace (e.g., [currency.USD](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USD) or [currency.USDT](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USDT)). The default is [syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency).

EXAMPLE

//**@version=**6

indicator("request.financial")

f = request.financial("NASDAQ:MSFT", "ACCOUNTS\_PAYABLE", "FY")

plot(f)

RETURNS

Requested series.

SEE ALSO

[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_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.

Some of the data previously provided by this function is available on TradingView through other feeds, such as "BCHAIN" or "FRED". Use Symbol Search to look for such data based on its description. Commitment of Traders (COT) data can be requested using the official [LibraryCOT](https://www.tradingview.com/v/ysFf2OTq/) library.

Requests [Nasdaq Data Link](https://data.nasdaq.com/) (formerly Quandl) data for a symbol.

SYNTAX

request.quandl(ticker, gaps, index, ignore\_invalid\_symbol) → series float

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".

**gaps (simple barmerge\_gaps)** Merge strategy for the requested data (requested data automatically merges with the main series: OHLC data). Possible values include: [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on), [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off). [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on) - requested data is merged with possible gaps ([na](https://www.tradingview.com/pine-script-reference/v6/#var_na) values). [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off) - requested data is merged continuously without gaps, all the gaps are filled with the previous, nearest existing values. Default value is [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off).

**index (series int)** A Quandl time-series column index.

**ignore\_invalid\_symbol (input bool)** An optional parameter. Determines the behavior of the function if the specified symbol is not found: if false, the script will halt and return a runtime error; if true, the function will return na and execution will continue. The default value is false.

EXAMPLE

//**@version=**6

indicator("request.quandl")

f = request.quandl("CFTC/SB\_FO\_ALL", barmerge.gaps\_off, 0)

plot(f)

RETURNS

Requested series.

REMARKS

You can learn more about how to find ticker and index values in our [Help Center](https://www.tradingview.com/chart/?solution=43000568613).

SEE ALSO

[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)

**request.security()**

Requests the result of an expression from a specified context (symbol and timeframe).

SYNTAX

request.security(symbol, timeframe, expression, gaps, lookahead, ignore\_invalid\_symbol, currency, calc\_bars\_count) → series <type>

ARGUMENTS

**symbol (series string)** Symbol or ticker identifier of the requested data. Use an empty string or [syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_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.

**timeframe (series string)** Timeframe of the requested data. Use an empty string or [timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period) to request data from the chart's timeframe or the timeframe specified in the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) function. To request data from a different timeframe, supply a valid timeframe string. See [here](https://www.tradingview.com/pine-script-docs/concepts/timeframes/#timeframe-string-specifications) to learn about specifying timeframe strings.

**expression (variable, function, object, array, matrix, or map of series int/float/bool/string/color/enum, or a tuple of these)** The expression to calculate and return from the requested context. It can accept a built-in variable like [close](https://www.tradingview.com/pine-script-reference/v6/#var_close), a user-defined variable, an expression such as ta.change(close) / (high - low), a function call that does not use Pine Script® drawings, an [object](https://www.tradingview.com/pine-script-docs/language/objects/), a [collection](https://www.tradingview.com/pine-script-docs/language/type-system/#collections), or a tuple of expressions.

**gaps (simple barmerge\_gaps)** Specifies how the returned values are merged on chart bars. Possible values: [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on), [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off). With [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on) a value only appears on the current chart bar when it first becomes available from the function's context, otherwise [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) is returned (thus a "gap" occurs). With [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off) what would otherwise be gaps are filled with the latest known value returned, avoiding [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) values. Optional. The default is [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off).

**lookahead (simple barmerge\_lookahead)** On historical bars only, returns data from the timeframe before it elapses. Possible values: [barmerge.lookahead\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_on), [barmerge.lookahead\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_off). Has no effect on realtime values. Optional. The default is [barmerge.lookahead\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_off) starting from Pine Script® v3. The default is [barmerge.lookahead\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_on) in v1 and v2. WARNING: Using [barmerge.lookahead\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_on) at timeframes higher than the chart's without offsetting the expression argument like in close[1] will introduce future leak in scripts, as the function will then return the close price before it is actually known in the current context. As is explained in the User Manual's page on [Repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/#future-leak-with-request-security) this will produce misleading results.

**ignore\_invalid\_symbol (input bool)** Determines the behavior of the function if the specified symbol is not found: if [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), the script will halt and throw a runtime error; if [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the function will return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) and execution will continue. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**currency (series string)** Optional. Specifies the target currency for converting values expressed in currency units (e.g., [open](https://www.tradingview.com/pine-script-reference/v6/#var_open), [high](https://www.tradingview.com/pine-script-reference/v6/#var_high), [low](https://www.tradingview.com/pine-script-reference/v6/#var_low), [close](https://www.tradingview.com/pine-script-reference/v6/#var_close)) or expressions involving such values. Literal values such as 200 are not converted. The conversion rate for monetary values depends on the previous daily value of a corresponding currency pair from the most popular exchange. A spread symbol is used if no exchange provides the rate directly. Possible values: a "string" representing a valid currency code (e.g., "USD" or "USDT") or a constant from the currency.\* namespace (e.g., [currency.USD](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USD) or [currency.USDT](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USDT)). The default is [syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency).

**calc\_bars\_count (simple int)** If specified, the function will only request this number of values from the end of the symbol's history and calculate expression as if these values are the only available data, which might improve calculation speed in some cases. Optional. The default is 100,000, which is the limit for all non-professional TradingView plans.

EXAMPLE

//**@version=**6

indicator("Simple `request.security()` calls")

// Returns 1D close of the current symbol.

dailyClose = request.security(syminfo.tickerid, "1D", close)

plot(dailyClose)

// Returns the close of "AAPL" from the same timeframe as currently open on the chart.

aaplClose = request.security("AAPL", timeframe.period, close)

plot(aaplClose)

EXAMPLE

//**@version=**6

indicator("Advanced `request.security()` calls")

// This calculates a 10-period moving average on the active chart.

sma = ta.sma(close, 10)

// This sends the `sma` calculation for execution in the context of the "AAPL" symbol at a "240" (4 hours) timeframe.

aaplSma = request.security("AAPL", "240", sma)

plot(aaplSma)

// To avoid differences on historical and realtime bars, you can use this technique, which only returns a value from the higher timeframe on the bar after it completes:

indexHighTF = barstate.isrealtime ? 1 : 0

indexCurrTF = barstate.isrealtime ? 0 : 1

nonRepaintingClose = request.security(syminfo.tickerid, "1D", close[indexHighTF])[indexCurrTF]

plot(nonRepaintingClose, "Non-repainting close")

// Returns the 1H close of "AAPL", extended session included. The value is dividend-adjusted.

extendedTicker = ticker.modify("NASDAQ:AAPL", session = session.extended, adjustment = adjustment.dividends)

aaplExtAdj = request.security(extendedTicker, "60", close)

plot(aaplExtAdj)

// Returns the result of a user-defined function.

// The `max` variable is mutable, but we can pass it to `request.security()` because it is wrapped in a function.

allTimeHigh(source) =>

var max = source

max := math.max(max, source)

allTimeHigh1D = request.security(syminfo.tickerid, "1D", allTimeHigh(high))

// By using a tuple `expression`, we obtain several values with only one `request.security()` call.

[open1D, high1D, low1D, close1D, ema1D] = request.security(syminfo.tickerid, "1D", [open, high, low, close, ta.ema(close, 10)])

plotcandle(open1D, high1D, low1D, close1D)

plot(ema1D)

// Returns an array containing the OHLC values of the chart's symbol from the 1D timeframe.

ohlcArray = request.security(syminfo.tickerid, "1D", array.from(open, high, low, close))

plotcandle(array.get(ohlcArray, 0), array.get(ohlcArray, 1), array.get(ohlcArray, 2), array.get(ohlcArray, 3))

RETURNS

A result determined by expression.

REMARKS

Scripts using this function might calculate differently on historical and realtime bars, leading to [repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

A single script can contain no more than 40 unique request.\*() function calls. A call is unique only if it does not call the same function with the same arguments.

When using two calls to a request.\*() function to evaluate the same expression from the same context with different calc\_bars\_count values, the second call requests the same number of historical bars as the first. For example, if a script calls request.security("AAPL", "", close, calc\_bars\_count = 3) after it calls request.security("AAPL", "", close, calc\_bars\_count = 5), the second call also uses five bars of historical data, not three.

The symbol of a request.() call can be *inherited* if it is not specified precisely, i.e., if the symbol argument is an empty string or [syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid). Similarly, the timeframe of a request.() call can be inherited if the timeframe argument is an empty string or [timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period). These values are normally taken from the chart on which the script is running. However, if request.\*() function A is called from within the expression of request.\*() function B, then function A can inherit the values from function B. See [here](https://www.tradingview.com/pine-script-docs/concepts/other-timeframes-and-data/#nested-requests) for more information.

SEE ALSO

[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period)[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify)[request.security\_lower\_tf](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security_lower_tf)[request.dividends](https://www.tradingview.com/pine-script-reference/v6/#fun_request.dividends)[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings)[request.splits](https://www.tradingview.com/pine-script-reference/v6/#fun_request.splits)[request.financial](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#type_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

request.security\_lower\_tf(symbol, timeframe, expression, ignore\_invalid\_symbol, currency, ignore\_invalid\_timeframe, calc\_bars\_count) → array<type>

ARGUMENTS

**symbol (series string)** Symbol or ticker identifier of the requested data. Use an empty string or [syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_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.

**timeframe (series string)** Timeframe of the requested data. Use an empty string or [timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period) to request data from the chart's timeframe or the timeframe specified in the [indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator) function. To request data from a different timeframe, supply a valid timeframe string. See [here](https://www.tradingview.com/pine-script-docs/concepts/timeframes/#timeframe-string-specifications) to learn about specifying timeframe strings.

**expression (variable, object or function of series int/float/bool/string/color/enum, or a tuple of these)** The expression to calculate and return from the requested context. It can accept a built-in variable like [close](https://www.tradingview.com/pine-script-reference/v6/#var_close), a user-defined variable, an expression such as ta.change(close) / (high - low), a function call that does not use Pine Script® drawings, an [object](https://www.tradingview.com/pine-script-docs/language/objects/), or a tuple of expressions. [Collections](https://www.tradingview.com/pine-script-docs/language/type-system/#collections) are not allowed unless they are within the fields of an object

**ignore\_invalid\_symbol (series bool)** Determines the behavior of the function if the specified symbol is not found: if [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), the script will halt and throw a runtime error; if [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the function will return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) and execution will continue. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**currency (series string)** Optional. Specifies the target currency for converting values expressed in currency units (e.g., [open](https://www.tradingview.com/pine-script-reference/v6/#var_open), [high](https://www.tradingview.com/pine-script-reference/v6/#var_high), [low](https://www.tradingview.com/pine-script-reference/v6/#var_low), [close](https://www.tradingview.com/pine-script-reference/v6/#var_close)) or expressions involving such values. Literal values such as 200 are not converted. The conversion rate for monetary values depends on the previous daily value of a corresponding currency pair from the most popular exchange. A spread symbol is used if no exchange provides the rate directly. Possible values: a "string" representing a valid currency code (e.g., "USD" or "USDT") or a constant from the currency.\* namespace (e.g., [currency.USD](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USD) or [currency.USDT](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USDT)). The default is [syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency).

**ignore\_invalid\_timeframe (series bool)** Determines the behavior of the function when the chart's timeframe is smaller than the timeframe used in the function call. If [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), the script will halt and throw a runtime error. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the function will return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) and execution will continue. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**calc\_bars\_count (simple int)** If specified, the function will only request this number of values from the end of the symbol's history and calculate expression as if these values are the only available data, which might improve calculation speed in some cases. Optional. The default is 100,000, which is the limit for all non-professional TradingView plans.

EXAMPLE

//**@version=**6

indicator("`request.security\_lower\_tf()` Example", overlay = true)

// If the current chart timeframe is set to 120 minutes, then the `arrayClose` array will contain two 'close' values from the 60 minute timeframe for each bar.

arrClose = request.security\_lower\_tf(syminfo.tickerid, "60", close)

if bar\_index == last\_bar\_index - 1

label.new(bar\_index, high, str.tostring(arrClose))

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](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

Please note that spreads (e.g., "AAPL+MSFT\*TSLA") do not always return reliable data with this function.

A single script can contain no more than 40 unique request.\*() function calls. A call is unique only if it does not call the same function with the same arguments.

When using two calls to a request.\*() function to evaluate the same expression from the same context with different calc\_bars\_count values, the second call requests the same number of historical bars as the first. For example, if a script calls request.security("AAPL", "", close, calc\_bars\_count = 3) after it calls request.security("AAPL", "", close, calc\_bars\_count = 5), the second call also uses five bars of historical data, not three.

The symbol of a request.() call can be *inherited* if it is not specified precisely, i.e., if the symbol argument is an empty string or [syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid). Similarly, the timeframe of a request.() call can be inherited if the timeframe argument is an empty string or [timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period). These values are normally taken from the chart that the script is running on. However, if request.\*() function A is called from within the expression of request.\*() function B, then function A can inherit the values from function B. See [here](https://www.tradingview.com/pine-script-docs/concepts/other-timeframes-and-data/#nested-requests) for more information.

SEE ALSO

[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period)[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)[request.dividends](https://www.tradingview.com/pine-script-reference/v6/#fun_request.dividends)[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings)[request.splits](https://www.tradingview.com/pine-script-reference/v6/#fun_request.splits)[request.financial](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://github.com/tradingview-eod/pine-seeds-docs).

SYNTAX

request.seed(source, symbol, expression, ignore\_invalid\_symbol, calc\_bars\_count) → series <type>

ARGUMENTS

**source (series string)** Name of the GitHub repository.

**symbol (series string)** Name of the file in the GitHub repository containing the data. The ".csv" file extension must not be included.

**expression (<arg\_expr\_type>)** An expression to be calculated and returned from the requested symbol's context. It can be a built-in variable like [close](https://www.tradingview.com/pine-script-reference/v6/#var_close), an expression such as ta.sma(close, 100), a non-mutable variable previously calculated in the script, a function call that does not use Pine Script® drawings, an array, a matrix, or a tuple. Mutable variables are not allowed, unless they are enclosed in the body of a function used in the expression.

**ignore\_invalid\_symbol (input bool)** Determines the behavior of the function if the specified symbol is not found: if [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), the script will halt and throw a runtime error; if [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the function will return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) and execution will continue. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**calc\_bars\_count (simple int)** If specified, the function will only request this number of values from the end of the symbol's history and calculate expression as if these values are the only available data, which might improve calculation speed in some cases. Optional. The default is 100,000, which is the limit for all non-professional TradingView plans.

EXAMPLE

//**@version=**6

indicator("BTC Development Activity")

[devAct, devActSMA] = request.seed("seed\_crypto\_santiment", "BTC\_DEV\_ACTIVITY", [close, ta.sma(close, 10)])

plot(devAct, "BTC Development Activity")

plot(devActSMA, "BTC Development Activity SMA10", color = color.yellow)

RETURNS

Requested series or tuple of series, which may include array/matrix IDs.

**request.splits()**

Requests splits data for the specified symbol.

SYNTAX

request.splits(ticker, field, gaps, lookahead, ignore\_invalid\_symbol) → series float

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](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker) will cause an error. Use [syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid) instead.

**field (series string)** Input string. Possible values include: [splits.denominator](https://www.tradingview.com/pine-script-reference/v6/#const_splits.denominator), [splits.numerator](https://www.tradingview.com/pine-script-reference/v6/#const_splits.numerator).

**gaps (simple barmerge\_gaps)** Merge strategy for the requested data (requested data automatically merges with the main series OHLC data). Possible values: [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on), [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off). [barmerge.gaps\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_on) - requested data is merged with possible gaps ([na](https://www.tradingview.com/pine-script-reference/v6/#var_na) values). [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off) - requested data is merged continuously without gaps, all the gaps are filled with the previous nearest existing values. Default value is [barmerge.gaps\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.gaps_off).

**lookahead (simple barmerge\_lookahead)** Merge strategy for the requested data position. Possible values: [barmerge.lookahead\_on](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_on), [barmerge.lookahead\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_off). Default value is [barmerge.lookahead\_off](https://www.tradingview.com/pine-script-reference/v6/#const_barmerge.lookahead_off) starting from version 3. Note that behavour is the same on real-time, and differs only on history.

**ignore\_invalid\_symbol (input bool)** An optional parameter. Determines the behavior of the function if the specified symbol is not found: if false, the script will halt and return a runtime error; if true, the function will return na and execution will continue. The default value is false.

EXAMPLE

//**@version=**6

indicator("request.splits")

s1 = request.splits("NASDAQ:BELFA", splits.denominator)

plot(s1)

s2 = request.splits("NASDAQ:BELFA", splits.denominator, gaps=barmerge.gaps\_on, lookahead=barmerge.lookahead\_on)

plot(s2)

RETURNS

Requested series, or n/a if there is no splits data for the specified symbol.

SEE ALSO

[request.earnings](https://www.tradingview.com/pine-script-reference/v6/#fun_request.earnings)[request.dividends](https://www.tradingview.com/pine-script-reference/v6/#fun_request.dividends)[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)

**runtime.error()**

When called, causes a runtime error with the error message specified in the message argument.

SYNTAX

runtime.error(message) → void

ARGUMENTS

**message (series string)** Error message.

**second()**2 overloads

SYNTAX & OVERLOADS

[second(time) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_second-0)

[second(time, timezone) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_second-1)

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

[second](https://www.tradingview.com/pine-script-reference/v6/#var_second)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[year](https://www.tradingview.com/pine-script-reference/v6/#fun_year)[month](https://www.tradingview.com/pine-script-reference/v6/#fun_month)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#fun_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#fun_minute)

**str.contains()**3 overloads

Returns true if the source string contains the str substring, false otherwise.

SYNTAX & OVERLOADS

[str.contains(source, str) → const bool](https://www.tradingview.com/pine-script-reference/v6/#fun_str.contains-0)

[str.contains(source, str) → simple bool](https://www.tradingview.com/pine-script-reference/v6/#fun_str.contains-1)

[str.contains(source, str) → series bool](https://www.tradingview.com/pine-script-reference/v6/#fun_str.contains-2)

ARGUMENTS

**source (const string)** Source string.

**str (const string)** The substring to search for.

EXAMPLE

//**@version=**6

indicator("str.contains")

// If the current chart is a continuous futures chart, e.g “BTC1!”, then the function will return true, false otherwise.

var isFutures = str.contains(syminfo.tickerid, "!")

plot(isFutures ? 1 : 0)

RETURNS

True if the str was found in the source string, false otherwise.

SEE ALSO

[str.pos](https://www.tradingview.com/pine-script-reference/v6/#fun_str.pos)[str.match](https://www.tradingview.com/pine-script-reference/v6/#fun_str.match)

**str.endswith()**3 overloads

Returns true if the source string ends with the substring specified in str, false otherwise.

SYNTAX & OVERLOADS

[str.endswith(source, str) → const bool](https://www.tradingview.com/pine-script-reference/v6/#fun_str.endswith-0)

[str.endswith(source, str) → simple bool](https://www.tradingview.com/pine-script-reference/v6/#fun_str.endswith-1)

[str.endswith(source, str) → series bool](https://www.tradingview.com/pine-script-reference/v6/#fun_str.endswith-2)

ARGUMENTS

**source (const string)** Source string.

**str (const string)** The substring to search for.

RETURNS

True if the source string ends with the substring specified in str, false otherwise.

SEE ALSO

[str.startswith](https://www.tradingview.com/pine-script-reference/v6/#fun_str.startswith)

**str.format()**2 overloads

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

[str.format(formatString, arg0, arg1, ...) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.format-0)

[str.format(formatString, arg0, arg1, ...) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.format-1)

ARGUMENTS

**formatString (simple string)** Format string.

**arg0, arg1, ... (simple int/float/bool/string)** Values to format.

EXAMPLE

//**@version=**6

indicator("str.format", overlay=true)

// The format specifier inside the curly braces accepts certain modifiers:

// - Specify the number of decimals to display:

s1 = str.format("{0,number,#.#}", 1.34) // returns: 1.3

label.new(bar\_index, close, text=s1)

// - Round a float value to an integer:

s2 = str.format("{0,number,integer}", 1.34) // returns: 1

label.new(bar\_index - 1, close, text=s2)

// - Display a number in currency:

s3 = str.format("{0,number,currency}", 1.34) // returns: $1.34

label.new(bar\_index - 2, close, text=s3)

// - Display a number as a percentage:

s4 = str.format("{0,number,percent}", 0.5) // returns: 50%

label.new(bar\_index - 3, close, text=s4)

// EXAMPLES WITH SEVERAL ARGUMENTS

// returns: Number 1 is not equal to 4

s5 = str.format("Number {0} is not {1} to {2}", 1, "equal", 4)

label.new(bar\_index - 4, close, text=s5)

// returns: 1.34 != 1.3

s6 = str.format("{0} != {0, number, #.#}", 1.34)

label.new(bar\_index - 5, close, text=s6)

// returns: 1 is equal to 1, but 2 is equal to 2

s7 = str.format("{0, number, integer} is equal to 1, but {1, number, integer} is equal to 2", 1.34, 1.52)

label.new(bar\_index - 6, close, text=s7)

// returns: The cash turnover amounted to $1,340,000.00

s8 = str.format("The cash turnover amounted to {0, number, currency}", 1340000)

label.new(bar\_index - 7, close, text=s8)

// returns: Expected return is 10% - 20%

s9 = str.format("Expected return is {0, number, percent} - {1, number, percent}", 0.1, 0.2)

label.new(bar\_index - 8, close, text=s9)

RETURNS

The formatted string.

REMARKS

By default, formatted numbers will display up to three decimals with no trailing zeros.

The string used as the formatString argument can contain single quote characters ('). However, one must pair all single quotes in that string to avoid unexpected formatting results.

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.

**str.format\_time()**

Converts the time timestamp into a string formatted according to format and timezone.

SYNTAX

str.format\_time(time, format, timezone) → series string

ARGUMENTS

**time (series int)** UNIX time, in milliseconds.

**format (series string)** A format string specifying the date/time representation of the time in the returned string. All letters used in the string, except those escaped by single quotation marks ', are considered formatting tokens and will be used as a formatting instruction. Refer to the Remarks section for a list of the most useful tokens. Optional. The default is "yyyy-MM-dd'T'HH:mm:ssZ", which represents the ISO 8601 standard.

**timezone (series string)** Allows adjusting the returned value to a time zone specified in either UTC/GMT notation (e.g., "UTC-5", "GMT+0530") or as an [IANA time zone database name](https://en.wikipedia.org/wiki/List_of_tz_database_time_zones) (e.g., "America/New\_York"). Optional. The default is [syminfo.timezone](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.timezone).

EXAMPLE

//**@version=**6

indicator("str.format\_time")

if timeframe.change("1D")

formattedTime = str.format\_time(time, "yyyy-MM-dd HH:mm", syminfo.timezone)

label.new(bar\_index, high, formattedTime)

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.

The most frequently used formatting tokens are:

y - Year. Use yy to output the last two digits of the year or yyyy to output all four. Year 2000 will be 00 with yy or 2000 with yyyy.

M - Month. Not to be confused with lowercase m, which stands for minute.

d - Day of the month.

a - AM/PM postfix.

h - Hour in the 12-hour format. The last hour of the day will be 11 in this format.

H - Hour in the 24-hour format. The last hour of the day will be 23 in this format.

m - Minute.

s - Second.

S - Fractions of a second.

Z - Timezone, the HHmm offset from UTC, preceded by either + or -.

**str.length()**3 overloads

Returns an integer corresponding to the amount of chars in that string.

SYNTAX & OVERLOADS

[str.length(string) → const int](https://www.tradingview.com/pine-script-reference/v6/#fun_str.length-0)

[str.length(string) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_str.length-1)

[str.length(string) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_str.length-2)

ARGUMENTS

**string (const string)** Source string.

RETURNS

The number of chars in source string.

**str.lower()**3 overloads

Returns a new string with all letters converted to lowercase.

SYNTAX & OVERLOADS

[str.lower(source) → const string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.lower-0)

[str.lower(source) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.lower-1)

[str.lower(source) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.lower-2)

ARGUMENTS

**source (const string)** String to be converted.

RETURNS

A new string with all letters converted to lowercase.

SEE ALSO

[str.upper](https://www.tradingview.com/pine-script-reference/v6/#fun_str.upper)

**str.match()**2 overloads

Returns the new substring of the source string if it matches a regex regular expression, an empty string otherwise.

SYNTAX & OVERLOADS

[str.match(source, regex) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.match-0)

[str.match(source, regex) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.match-1)

ARGUMENTS

**source (simple string)** Source string.

**regex (simple string)** The regular expression to which this string is to be matched.

EXAMPLE

//**@version=**6

indicator("str.match")

s = input.string("It's time to sell some NASDAQ:AAPL!")

// finding first substring that matches regular expression "[\w]+:[\w]+"

var **string** tickerid = str.match(s, "[\\w]+:[\\w]+")

if barstate.islastconfirmedhistory

label.new(bar\_index, high, text = tickerid) // "NASDAQ:AAPL"

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](https://en.wikipedia.org/wiki/Regular_expression#Perl_and_PCRE) in the source string.

The backslash "\" symbol in theregex string needs to be escaped with additional backslash, e.g. "\\d" stands for regular expression "\d".

SEE ALSO

[str.contains](https://www.tradingview.com/pine-script-reference/v6/#fun_str.contains)[str.substring](https://www.tradingview.com/pine-script-reference/v6/#fun_str.substring)

**str.pos()**3 overloads

Returns the position of the first occurrence of the str string in the source string, 'na' otherwise.

SYNTAX & OVERLOADS

[str.pos(source, str) → const int](https://www.tradingview.com/pine-script-reference/v6/#fun_str.pos-0)

[str.pos(source, str) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_str.pos-1)

[str.pos(source, str) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_str.pos-2)

ARGUMENTS

**source (const string)** Source string.

**str (const string)** The substring to search for.

RETURNS

Position of the str string in the source string.

REMARKS

Strings indexing starts at 0.

SEE ALSO

[str.contains](https://www.tradingview.com/pine-script-reference/v6/#fun_str.contains)[str.match](https://www.tradingview.com/pine-script-reference/v6/#fun_str.match)[str.substring](https://www.tradingview.com/pine-script-reference/v6/#fun_str.substring)

**str.repeat()**4 overloads

Constructs a new string containing the source string repeated repeat times with the separator injected between each repeated instance.

SYNTAX & OVERLOADS

[str.repeat(source, repeat, separator) → const string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.repeat-0)

[str.repeat(source, repeat, separator) → input string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.repeat-1)

[str.repeat(source, repeat, separator) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.repeat-2)

[str.repeat(source, repeat, separator) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.repeat-3)

ARGUMENTS

**source (const string)** String to repeat.

**repeat (const int)** Number of times to repeat the source string. Must be greater than or equal to 0.

**separator (const string)** String to inject between repeated values. Optional. The default is empty string.

EXAMPLE

//**@version=**6

indicator("str.repeat")

repeat = str.repeat("?", 3, ",") // Returns "?,?,?"

label.new(bar\_index,close,repeat)

REMARKS

Returns [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) if the source is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**str.replace()**3 overloads

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

[str.replace(source, target, replacement, occurrence) → const string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.replace-0)

[str.replace(source, target, replacement, occurrence) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.replace-1)

[str.replace(source, target, replacement, occurrence) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.replace-2)

ARGUMENTS

**source (const string)** Source string.

**target (const string)** String to be replaced.

**replacement (const string)** String to be inserted instead of the target string.

**occurrence (const int)** N-th occurrence of the target string to replace. Indexing starts at 0 for the first match. Optional. Default value is 0.

EXAMPLE

//**@version=**6

indicator("str.replace")

var source = "FTX:BTCUSD / FTX:BTCEUR"

// Replace first occurrence of "FTX" with "BINANCE" replacement string

var newSource = str.replace(source, "FTX", "BINANCE", 0)

if barstate.islastconfirmedhistory

// Display "BINANCE:BTCUSD / FTX:BTCEUR"

label.new(bar\_index, high, text = newSource)

RETURNS

Processed string.

SEE ALSO

[str.replace\_all](https://www.tradingview.com/pine-script-reference/v6/#fun_str.replace_all)[str.match](https://www.tradingview.com/pine-script-reference/v6/#fun_str.match)

**str.replace\_all()**2 overloads

Replaces each occurrence of the target string in the source string with the replacement string.

SYNTAX & OVERLOADS

[str.replace\_all(source, target, replacement) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.replace_all-0)

[str.replace\_all(source, target, replacement) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.replace_all-1)

ARGUMENTS

**source (simple string)** Source string.

**target (simple string)** String to be replaced.

**replacement (simple string)** String to be substituted for each occurrence of target string.

RETURNS

Processed string.

**str.split()**

Divides a string into an array of substrings and returns its array id.

SYNTAX

str.split(string, separator) → array<string>

ARGUMENTS

**string (series string)** Source string.

**separator (series string)** The string separating each substring.

RETURNS

The id of an array of strings.

**str.startswith()**3 overloads

Returns true if the source string starts with the substring specified in str, false otherwise.

SYNTAX & OVERLOADS

[str.startswith(source, str) → const bool](https://www.tradingview.com/pine-script-reference/v6/#fun_str.startswith-0)

[str.startswith(source, str) → simple bool](https://www.tradingview.com/pine-script-reference/v6/#fun_str.startswith-1)

[str.startswith(source, str) → series bool](https://www.tradingview.com/pine-script-reference/v6/#fun_str.startswith-2)

ARGUMENTS

**source (const string)** Source string.

**str (const string)** The substring to search for.

RETURNS

True if the source string starts with the substring specified in str, false otherwise.

SEE ALSO

[str.endswith](https://www.tradingview.com/pine-script-reference/v6/#fun_str.endswith)

**str.substring()**6 overloads

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

[str.substring(source, begin\_pos) → const string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.substring-0)

[str.substring(source, begin\_pos) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.substring-1)

[str.substring(source, begin\_pos) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.substring-2)

[str.substring(source, begin\_pos, end\_pos) → const string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.substring-3)

[str.substring(source, begin\_pos, end\_pos) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.substring-4)

[str.substring(source, begin\_pos, end\_pos) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.substring-5)

ARGUMENTS

**source (const string)** Source string from which to extract the substring.

**begin\_pos (const int)** The beginning position of the extracted substring. It is inclusive (the extracted substring includes the character at that position).

EXAMPLE

//**@version=**6

indicator("str.substring", overlay = true)

sym= input.symbol("NASDAQ:AAPL")

pos = str.pos(sym, ":") // Get position of ":" character

tkr= str.substring(sym, pos+1) // "AAPL"

if barstate.islastconfirmedhistory

label.new(bar\_index, high, text = tkr)

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

[str.contains](https://www.tradingview.com/pine-script-reference/v6/#fun_str.contains)[str.pos](https://www.tradingview.com/pine-script-reference/v6/#fun_str.pos)[str.match](https://www.tradingview.com/pine-script-reference/v6/#fun_str.match)

**str.tonumber()**4 overloads

Converts a value represented in string to its "float" equivalent.

SYNTAX & OVERLOADS

[str.tonumber(string) → const float](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tonumber-0)

[str.tonumber(string) → input float](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tonumber-1)

[str.tonumber(string) → simple float](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tonumber-2)

[str.tonumber(string) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tonumber-3)

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](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**str.tostring()**4 overloads

SYNTAX & OVERLOADS

[str.tostring(value, format) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tostring-0)

[str.tostring(value, format) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tostring-1)

[str.tostring(value) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tostring-2)

[str.tostring(value) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tostring-3)

ARGUMENTS

**value (simple int/float)** Value or array ID whose elements are converted to a string.

**format (simple string)** Format string. Accepts these format.\* constants: [format.mintick](https://www.tradingview.com/pine-script-reference/v6/#const_format.mintick), [format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent), [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume). Optional. The default value is '#.##########'.

RETURNS

The string representation of the value argument.

If the value argument is a string, it is returned as is.

When the value is na, the function returns the string "NaN".

REMARKS

The formatting of float values will also round those values when necessary, e.g. str.tostring(3.99, '#') will return "4".

To display trailing zeros, use '0' instead of '#'. For example, '#.000'.

When using [format.mintick](https://www.tradingview.com/pine-script-reference/v6/#const_format.mintick), the value will be rounded to the nearest number that can be divided by [syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mintick) without the remainder. The string is returned with trailing zeros.

If the x argument is a string, the same string value will be returned.

Bool type arguments return "true" or "false".

When x is na, the function returns "NaN".

**str.trim()**4 overloads

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

[str.trim(source) → const string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.trim-0)

[str.trim(source) → input string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.trim-1)

[str.trim(source) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.trim-2)

[str.trim(source) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.trim-3)

ARGUMENTS

**source (const string)** String to trim.

EXAMPLE

//**@version=**6

indicator("str.trim")

trim = str.trim(" abc ") // Returns "abc"

label.new(bar\_index,close,trim)

REMARKS

Returns an empty string ("") if the result is empty after the trim or if the source is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**str.upper()**3 overloads

Returns a new string with all letters converted to uppercase.

SYNTAX & OVERLOADS

[str.upper(source) → const string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.upper-0)

[str.upper(source) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.upper-1)

[str.upper(source) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_str.upper-2)

ARGUMENTS

**source (const string)** String to be converted.

RETURNS

A new string with all letters converted to uppercase.

SEE ALSO

[str.lower](https://www.tradingview.com/pine-script-reference/v6/#fun_str.lower)

**strategy()**

This declaration statement designates the script as a strategy and sets a number of strategy-related properties.

SYNTAX

strategy(title, shorttitle, overlay, format, precision, scale, pyramiding, calc\_on\_order\_fills, calc\_on\_every\_tick, max\_bars\_back, backtest\_fill\_limits\_assumption, default\_qty\_type, default\_qty\_value, initial\_capital, currency, slippage, commission\_type, commission\_value, process\_orders\_on\_close, close\_entries\_rule, margin\_long, margin\_short, explicit\_plot\_zorder, max\_lines\_count, max\_labels\_count, max\_boxes\_count, calc\_bars\_count, risk\_free\_rate, use\_bar\_magnifier, fill\_orders\_on\_standard\_ohlc, max\_polylines\_count, dynamic\_requests, behind\_chart) → void

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.

**shorttitle (const string)** The script's display name on charts. If specified, it will replace the title argument in most chart-related windows. Optional. The default is the argument used for title.

**overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the strategy will be displayed over the chart. If [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), it will be added in a separate pane. Strategy-specific labels that display entries and exits will be displayed over the main chart regardless of this setting. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**format (const string)** Specifies the formatting of the script's displayed values. Possible values: [format.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_format.inherit), [format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price), [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume), [format.percent](https://www.tradingview.com/pine-script-reference/v6/#const_format.percent). Optional. The default is [format.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_format.inherit).

**precision (const int)** Specifies the number of digits after the floating point of the script's displayed values. Must be a non-negative integer no greater than 16. If format is set to [format.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_format.inherit) and precision is specified, the format will instead be set to [format.price](https://www.tradingview.com/pine-script-reference/v6/#const_format.price). When the function's format parameter uses [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume), the precision parameter will not affect the result, as the decimal precision rules defined by [format.volume](https://www.tradingview.com/pine-script-reference/v6/#const_format.volume) supersede other precision settings. Optional. The default is inherited from the precision of the chart's symbol.

**scale (const scale\_type)** The price scale used. Possible values: [scale.right](https://www.tradingview.com/pine-script-reference/v6/#const_scale.right), [scale.left](https://www.tradingview.com/pine-script-reference/v6/#const_scale.left), [scale.none](https://www.tradingview.com/pine-script-reference/v6/#const_scale.none). The [scale.none](https://www.tradingview.com/pine-script-reference/v6/#const_scale.none) value can only be applied in combination with overlay = true. Optional. By default, the script uses the same scale as the chart.

**pyramiding (const int)** The maximum number of entries allowed in the same direction. If the value is 0, only one entry order in the same direction can be opened, and additional entry orders are rejected. This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. The default is 0.

**calc\_on\_order\_fills (const bool)** Specifies whether the strategy should be recalculated after an order is filled. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the strategy recalculates after an order is filled, as opposed to recalculating only when the bar closes. This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**calc\_on\_every\_tick (const bool)** Specifies whether the strategy should be recalculated on each realtime tick. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), when the strategy is running on a realtime bar, it will recalculate on each chart update. If [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), the strategy only calculates when the realtime bar closes. The argument used does not affect strategy calculation on historical data. This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**max\_bars\_back (const int)** The length of the historical buffer the script keeps for every variable and function, which determines how many past values can be referenced using the [] history-referencing operator. The required buffer size is automatically detected by the Pine Script® runtime. Using this parameter is only necessary when a runtime error occurs because automatic detection fails. More information on the underlying mechanics of the historical buffer can be found [in our Help Center](https://www.tradingview.com/chart/?solution=43000587849). Optional. The default is 0.

**backtest\_fill\_limits\_assumption (const int)** Limit order execution threshold in ticks. When it is used, limit orders are only filled if the market price exceeds the order's limit level by the specified number of ticks. Optional. The default is 0.

**default\_qty\_type (const string)** Specifies the units used for default\_qty\_value. Possible values are: [strategy.fixed](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.fixed) for contracts/shares/lots, [strategy.cash](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.cash) for currency amounts, or [strategy.percent\_of\_equity](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.percent_of_equity) for a percentage of available equity. This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. The default is [strategy.fixed](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.fixed).

**default\_qty\_value (const int/float)** The default quantity to trade, in units determined by the argument used with the default\_qty\_type parameter. This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. The default is 1.

**initial\_capital (const int/float)** The amount of funds initially available for the strategy to trade, in units of currency. Optional. The default is 1000000.

**currency (const string)** Currency used by the strategy in currency-related calculations. Market positions are still opened by converting currency into the chart symbol's currency. The conversion rate depends on the previous daily value of a corresponding currency pair from the most popular exchange. A spread symbol is used if no exchange provides the rate directly. Possible values: a "string" representing a valid currency code (e.g., "USD" or "USDT") or a constant from the currency.\* namespace (e.g., [currency.USD](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USD) or [currency.USDT](https://www.tradingview.com/pine-script-reference/v6/#const_currency.USDT)). The default is [syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency).

**slippage (const int)** Slippage expressed in ticks. This value is added to or subtracted from the fill price of market/stop orders to make the fill price less favorable for the strategy. E.g., if [syminfo.mintick](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.mintick) is 0.01 and slippage is set to 5, a long market order will enter at 5 \* 0.01 = 0.05 points above the actual price. This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. The default is 0.

**commission\_type (const string)** Determines what the number passed to the commission\_value expresses: [strategy.commission.percent](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.commission.percent) for a percentage of the cash volume of the order, [strategy.commission.cash\_per\_contract](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.commission.cash_per_contract) for currency per contract, [strategy.commission.cash\_per\_order](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.commission.cash_per_order) for currency per order. This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. The default is [strategy.commission.percent](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.commission.percent).

**commission\_value (const int/float)** Commission applied to the strategy's orders in units determined by the argument passed to the commission\_type parameter. This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. The default is 0.

**process\_orders\_on\_close (const bool)** When set to [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), generates an additional attempt to execute orders after a bar closes and strategy calculations are completed. If the orders are market orders, the broker emulator executes them before the next bar's open. If the orders are price-dependent, they will only be filled if the price conditions are met. This option is useful if you wish to close positions on the current bar. This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**close\_entries\_rule (const string)** Determines the order in which trades are closed. Possible values are: "FIFO" (First-In, First-Out) if the earliest exit order must close the earliest entry order, or "ANY" if the orders are closed based on the from\_entry parameter of the [strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit) function. "FIFO" can only be used with stocks, futures and US forex (NFA Compliance Rule 2-43b), while "ANY" is allowed in non-US forex. Optional. The default is "FIFO".

**margin\_long (const int/float)** Margin long is the percentage of the purchase price of a security that must be covered by cash or collateral for long positions. Must be a non-negative number. The logic used to simulate margin calls is explained in the [Help Center](https://www.tradingview.com/chart/?solution=43000628599). This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. If the value is 0, the strategy does not enforce any limits on position size. The default is 100, in which case the strategy only uses its own funds and the long positions cannot be margin called.

**margin\_short (const int/float)** Margin short is the percentage of the purchase price of a security that must be covered by cash or collateral for short positions. Must be a non-negative number. The logic used to simulate margin calls is explained in the [Help Center](https://www.tradingview.com/chart/?solution=43000628599). This setting can also be changed in the strategy's "Settings/Properties" tab. Optional. If the value is 0, the strategy does not enforce any limits on position size. The default is 100, in which case the strategy only uses its own funds. Note that even with no margin used, short positions *can* be margin called if the loss exceeds available funds.

**explicit\_plot\_zorder (const bool)** Specifies the order in which the script's plots, fills, and hlines are rendered. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), plots are drawn in the order in which they appear in the script's code, each newer plot being drawn above the previous ones. This only applies to plot\*() functions, [fill](https://www.tradingview.com/pine-script-reference/v6/#fun_fill), and [hline](https://www.tradingview.com/pine-script-reference/v6/#fun_hline). Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**max\_lines\_count (const int)** The number of last [line](https://www.tradingview.com/pine-script-reference/v6/#type_line) drawings displayed. Possible values: 1-500. Optional. The default is 50.

**max\_labels\_count (const int)** The number of last [label](https://www.tradingview.com/pine-script-reference/v6/#type_label) drawings displayed. Possible values: 1-500. Optional. The default is 50.

**max\_boxes\_count (const int)** The number of last [box](https://www.tradingview.com/pine-script-reference/v6/#type_box) drawings displayed. Possible values: 1-500. Optional. The default is 50.

**calc\_bars\_count (const int)** Limits the initial calculation of a script to the last number of bars specified. When specified, a "Calculated bars" field will be included in the "Calculation" section of the script's "Settings/Inputs" tab. Optional. The default is 0, in which case the script executes on all available bars.

**risk\_free\_rate (const int/float)** The risk-free rate of return is the annual percentage change in the value of an investment with minimal or zero risk. It is used to calculate the [Sharpe](https://www.tradingview.com/support/solutions/43000681694) and [Sortino](https://www.tradingview.com/support/solutions/43000681697) ratios. Optional. The default is 2.

**use\_bar\_magnifier (const bool)** Optional. When [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the [Broker Emulator](https://www.tradingview.com/pine-script-docs/concepts/strategies/#broker-emulator) uses lower timeframe data during backtesting on historical bars to achieve more realistic results. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false). Only [Premium](https://www.tradingview.com/gopro/) and higher-tier plans have access to this feature.

**fill\_orders\_on\_standard\_ohlc (const bool)** When [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), forces strategies running on Heikin Ashi charts to fill orders using actual OHLC prices, for more realistic results. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**max\_polylines\_count (const int)** The number of last [polyline](https://www.tradingview.com/pine-script-reference/v6/#type_polyline) drawings displayed. Possible values: 1-100. The count is approximate; more drawings than the specified count may be displayed. Optional. The default is 50.

**dynamic\_requests (const bool)** Specifies whether the script can dynamically call functions from the request.\*() namespace. Dynamic request.\*() calls are allowed within the local scopes of conditional structures (e.g., [if](https://www.tradingview.com/pine-script-reference/v6/#kw_if)), loops (e.g., [for](https://www.tradingview.com/pine-script-reference/v6/#kw_for)), and exported functions. Additionally, such calls allow "series" arguments for many of their parameters. Optional. The default is [true](https://www.tradingview.com/pine-script-reference/v6/#const_true). See the User Manual's [Dynamic requests](https://www.tradingview.com/pine-script-docs/concepts/other-timeframes-and-data/#dynamic-requests) section for more information.

**behind\_chart (const bool)** Controls whether the script's plots and drawings in the main chart pane appear behind the chart display (if [true](https://www.tradingview.com/pine-script-reference/v6/#const_true)), or in front of it (if [false](https://www.tradingview.com/pine-script-reference/v6/#const_false)). Optional. The default is [true](https://www.tradingview.com/pine-script-reference/v6/#const_true).

EXAMPLE

//**@version=**6

strategy("My strategy", overlay = true)

// Enter long by market if current open is greater than previous high.

if open > high[1]

strategy.entry("Long", strategy.long, 1)

// Generate a full exit bracket (profit 10 points, loss 5 points per contract) from the entry named "Long".

strategy.exit("Exit", "Long", profit = 10, loss = 5)

REMARKS

You can learn more about strategies in our [User Manual](https://www.tradingview.com/pine-script-docs/concepts/strategies/).

Every strategy script must have one [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) call.

Strategies using calc\_on\_every\_tick = true parameter may calculate differently on historical and realtime bars, which causes [repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

Strategies always use the chart's prices to enter and exit positions. Using them on non-standard chart types (Heikin Ashi, Renko, etc.) will produce misleading results, as their prices are synthetic. Backtesting on non-standard charts is thus not recommended.

The maximum number of orders a strategy can open, unless it uses Deep Backtesting mode, is 9000. If the strategy exceeds this limit, it removes the oldest order's information when a new entry appears in the "List of Trades" tab. The strategy.closedtrades.\*() functions return [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) for trades opened or closed by removed orders. To retrieve the index of the oldest available closed trade, use the [strategy.closedtrades.first\_index](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.closedtrades.first_index) variable.

SEE ALSO

[indicator](https://www.tradingview.com/pine-script-reference/v6/#fun_indicator)[library](https://www.tradingview.com/pine-script-reference/v6/#fun_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.

This command is most useful when working with price-based orders (e.g., [limit orders](https://www.tradingview.com/pine-script-docs/concepts/strategies/#limit-orders)). Calls to this command can also cancel [market orders](https://www.tradingview.com/pine-script-docs/concepts/strategies/#market-orders), but only if they execute on the same ticks as the order placement commands.

SYNTAX

strategy.cancel(id) → void

ARGUMENTS

**id (series string)** The identifier of the unfilled order to cancel.

EXAMPLE

//**@version=**6

strategy(title = "Order cancellation demo")

conditionForBuy = open > high[1]

if conditionForBuy

strategy.entry("Long", strategy.long, 1, limit = low) // Enter long using limit order at low price of current bar if `conditionForBuy` is `true`.

if not conditionForBuy

strategy.cancel("Long") // Cancel the entry order with name "Long" if `conditionForBuy` is `false`.

**strategy.cancel\_all()**

Cancels all pending or unfilled orders, regardless of their identifiers.

This command is most useful when working with price-based orders (e.g., [limit orders](https://www.tradingview.com/pine-script-docs/concepts/strategies/#limit-orders)). Calls to this command can also cancel [market orders](https://www.tradingview.com/pine-script-docs/concepts/strategies/#market-orders), but only if they execute on the same ticks as the order placement commands.

SYNTAX

strategy.cancel\_all() → void

EXAMPLE

//**@version=**6

strategy(title = "Cancel all orders demo")

conditionForBuy1 = open > high[1]

if conditionForBuy1

strategy.entry("Long entry 1", strategy.long, 1, limit = low) // Enter long using a limit order if `conditionForBuy1` is `true`.

conditionForBuy2 = conditionForBuy1 and open[1] > high[2]

**float** lowest2 = ta.lowest(low, 2)

if conditionForBuy2

strategy.entry("Long entry 2", strategy.long, 1, limit = lowest2) // Enter long using a limit order if `conditionForBuy2` is `true`.

conditionForStopTrading = open < lowest2

if conditionForStopTrading

strategy.cancel\_all() // Cancel both limit orders if `conditionForStopTrading` is `true`.

**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.

This command always generates [market orders](https://www.tradingview.com/pine-script-docs/concepts/strategies/#market-orders). To exit from a position using price-based orders (e.g., [stop-loss](https://www.tradingview.com/pine-script-docs/concepts/strategies/#take-profit-and-stop-loss) orders), use the [strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit) command.

SYNTAX

strategy.close(id, comment, qty, qty\_percent, alert\_message, immediately, disable\_alert) → void

ARGUMENTS

**id (series string)** The entry identifier of the open trades to close.

**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.

**qty (series int/float)** Optional. The number of contracts/lots/shares/units to close when an exit order fills. If specified, the command uses this value instead of qty\_percent to determine the order size. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), which means the order size depends on the qty\_percent value.

**qty\_percent (series int/float)** Optional. A value between 0 and 100 representing the percentage of the open trade quantity to close when an exit order fills. The percentage calculation depends on the total size of the open trades with the id entry identifier. The command ignores this parameter if the qty value is not [na](https://www.tradingview.com/pine-script-reference/v6/#var_na). The default is 100.

**alert\_message (series string)** Optional. Custom text for the alert that fires when an order fills. If the "Message" field of the "Create Alert" dialog box contains the {{strategy.order.alert\_message}} placeholder, the alert message replaces the placeholder with this text. The default is an empty string.

**immediately (series bool)** Optional. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the closing order executes on the same tick when the strategy places it, ignoring the strategy properties that restrict execution to the opening tick of the following bar. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

**disable\_alert (series bool)** Optional. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) when the command creates an order, the strategy does not trigger an alert when that order fills. This parameter accepts a "series" value, meaning users can control which orders trigger alerts when they execute. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

strategy("Partial close strategy")

// Calculate a 14-bar and 28-bar moving average of `close` prices.

**float** sma14 = ta.sma(close, 14)

**float** sma28 = ta.sma(close, 28)

// Place a market order to enter a long position when `sma14` crosses over `sma28`.

if ta.crossover(sma14, sma28)

strategy.entry("My Long Entry ID", strategy.long)

// Place a market order to close the long trade when `sma14` crosses under `sma28`.

if ta.crossunder(sma14, sma28)

strategy.close("My Long Entry ID", "50% market close", qty\_percent = 50)

// Plot the position size.

plot(strategy.position\_size)

REMARKS

When a position consists of several open trades and the close\_entries\_rule in the [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) declaration statement is "FIFO" (default), a [strategy.close](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-docs/concepts/strategies/#closing-a-market-position) section of our User Manual.

**strategy.close\_all()**2 overloads

Creates an order to close an open position completely, regardless of the identifiers of the entry orders that opened or added to it.

This command always generates [market orders](https://www.tradingview.com/pine-script-docs/concepts/strategies/#market-orders). To exit from a position using price-based orders (e.g., [stop-loss](https://www.tradingview.com/pine-script-docs/concepts/strategies/#take-profit-and-stop-loss) orders), use the [strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit) command.

SYNTAX & OVERLOADS

[strategy.close\_all(comment, alert\_message) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.close_all-0)

[strategy.close\_all(comment, alert\_message, immediately, disable\_alert) → void](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.close_all-1)

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.

**alert\_message (series string)** Optional. Custom text for the alert that fires when an order fills. If the "Message" field of the "Create Alert" dialog box contains the {{strategy.order.alert\_message}} placeholder, the alert message replaces the placeholder with this text. The default is an empty string.

EXAMPLE

//**@version=**6

strategy("Multi-entry close strategy")

// Calculate a 14-bar and 28-bar moving average of `close` prices.

**float** sma14 = ta.sma(close, 14)

**float** sma28 = ta.sma(close, 28)

// Place a market order to enter a long trade every time `sma14` crosses over `sma28`.

if ta.crossover(sma14, sma28)

strategy.order("My Long Entry ID " + str.tostring(strategy.opentrades), strategy.long)

// Place a market order to close the entire position every 500 bars.

if bar\_index % 500 == 0

strategy.close\_all()

// Plot the position size.

plot(strategy.position\_size)

**strategy.closedtrades.commission()**

Returns the sum of entry and exit fees paid in the closed trade, expressed in [strategy.account\_currency](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency).

SYNTAX

strategy.closedtrades.commission(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("`strategy.closedtrades.commission` Example", commission\_type = strategy.commission.percent, commission\_value = 0.1)

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Plot total fees for the latest closed trade.

plot(strategy.closedtrades.commission(strategy.closedtrades - 1))

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)[strategy.opentrades.commission](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.commission)

**strategy.closedtrades.entry\_bar\_index()**

Returns the bar\_index of the closed trade's entry.

SYNTAX

strategy.closedtrades.entry\_bar\_index(trade\_num) → series int

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.closedtrades.entry\_bar\_index Example")

// Enter long trades on three rising bars; exit on two falling bars.

if ta.rising(close, 3)

strategy.entry("Long", strategy.long)

if ta.falling(close, 2)

strategy.close("Long")

// Function that calculates the average amount of bars in a trade.

avgBarsPerTrade() =>

sumBarsPerTrade = 0

for tradeNo = 0 to strategy.closedtrades - 1

// Loop through all closed trades, starting with the oldest.

sumBarsPerTrade += strategy.closedtrades.exit\_bar\_index(tradeNo) - strategy.closedtrades.entry\_bar\_index(tradeNo) + 1

result = nz(sumBarsPerTrade / strategy.closedtrades)

plot(avgBarsPerTrade())

SEE ALSO

[strategy.closedtrades.exit\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.exit_bar_index)[strategy.opentrades.entry\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.entry_bar_index)

**strategy.closedtrades.entry\_comment()**

Returns the comment message of the closed trade's entry, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) if there is no entry with this trade\_num.

SYNTAX

strategy.closedtrades.entry\_comment(trade\_num) → series string

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("`strategy.closedtrades.entry\_comment()` Example", overlay = true)

stopPrice = open \* 1.01

longCondition = ta.crossover(ta.sma(close, 14), ta.sma(close, 28))

if (longCondition)

strategy.entry("Long", strategy.long, stop = stopPrice, comment = str.tostring(stopPrice, "#.####"))

strategy.exit("EXIT", trail\_points = 1000, trail\_offset = 0)

var testTable = table.new(position.top\_right, 1, 3, color.orange, border\_width = 1)

if barstate.islastconfirmedhistory or barstate.isrealtime

table.cell(testTable, 0, 0, 'Last closed trade:')

table.cell(testTable, 0, 1, "Order stop price value: " + strategy.closedtrades.entry\_comment(strategy.closedtrades - 1))

table.cell(testTable, 0, 2, "Actual Entry Price: " + str.tostring(strategy.closedtrades.entry\_price(strategy.closedtrades - 1)))

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)[strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry)[strategy.closedtrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.closedtrades)

**strategy.closedtrades.entry\_id()**

Returns the id of the closed trade's entry.

SYNTAX

strategy.closedtrades.entry\_id(trade\_num) → series string

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.closedtrades.entry\_id Example", overlay = true)

// Enter a short position and close at the previous to last bar.

if bar\_index == 1

strategy.entry("Short at bar #" + str.tostring(bar\_index), strategy.short)

if bar\_index == last\_bar\_index - 2

strategy.close\_all()

// Display ID of the last entry position.

if barstate.islastconfirmedhistory

label.new(last\_bar\_index, high, "Last Entry ID is: " + strategy.closedtrades.entry\_id(strategy.closedtrades - 1))

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

[strategy.closedtrades.entry\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.entry_bar_index)[strategy.closedtrades.entry\_price](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.entry_price)[strategy.closedtrades.entry\_time](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.entry_time)

**strategy.closedtrades.entry\_price()**

Returns the price of the closed trade's entry.

SYNTAX

strategy.closedtrades.entry\_price(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.closedtrades.entry\_price Example 1")

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Return the entry price for the latest entry.

entryPrice = strategy.closedtrades.entry\_price(strategy.closedtrades - 1)

plot(entryPrice, "Long entry price")

EXAMPLE

// Calculates the average profit percentage for all closed trades.

//**@version=**6

strategy("strategy.closedtrades.entry\_price Example 2")

// Strategy calls to create single short and long trades

if bar\_index == last\_bar\_index - 15

strategy.entry("Long Entry", strategy.long)

else if bar\_index == last\_bar\_index - 10

strategy.close("Long Entry")

strategy.entry("Short", strategy.short)

else if bar\_index == last\_bar\_index - 5

strategy.close("Short")

// Calculate profit for both closed trades.

profitPct = 0.0

for tradeNo = 0 to strategy.closedtrades - 1

entryP = strategy.closedtrades.entry\_price(tradeNo)

exitP = strategy.closedtrades.exit\_price(tradeNo)

profitPct += (exitP - entryP) / entryP \* strategy.closedtrades.size(tradeNo) \* 100

// Calculate average profit percent for both closed trades.

avgProfitPct = nz(profitPct / strategy.closedtrades)

plot(avgProfitPct)

SEE ALSO

[strategy.closedtrades.entry\_price](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.entry_price)[strategy.closedtrades.exit\_price](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.exit_price)[strategy.closedtrades.size](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.size)[strategy.closedtrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.closedtrades)

**strategy.closedtrades.entry\_time()**

Returns the UNIX time of the closed trade's entry, expressed in milliseconds..

SYNTAX

strategy.closedtrades.entry\_time(trade\_num) → series int

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.closedtrades.entry\_time Example", overlay = true)

// Enter long trades on three rising bars; exit on two falling bars.

if ta.rising(close, 3)

strategy.entry("Long", strategy.long)

if ta.falling(close, 2)

strategy.close("Long")

// Calculate the average trade duration

avgTradeDuration() =>

sumTradeDuration = 0

for i = 0 to strategy.closedtrades - 1

sumTradeDuration += strategy.closedtrades.exit\_time(i) - strategy.closedtrades.entry\_time(i)

result = nz(sumTradeDuration / strategy.closedtrades)

// Display average duration converted to seconds and formatted using 2 decimal points

if barstate.islastconfirmedhistory

label.new(bar\_index, high, str.tostring(avgTradeDuration() / 1000, "#.##") + " seconds")

SEE ALSO

[strategy.opentrades.entry\_time](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.entry_time)[strategy.closedtrades.exit\_time](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.exit_time)[time](https://www.tradingview.com/pine-script-reference/v6/#var_time)

**strategy.closedtrades.exit\_bar\_index()**

Returns the bar\_index of the closed trade's exit.

SYNTAX

strategy.closedtrades.exit\_bar\_index(trade\_num) → series int

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.closedtrades.exit\_bar\_index Example 1")

// Strategy calls to place a single short trade. We enter the trade at the first bar and exit the trade at 10 bars before the last chart bar.

if bar\_index == 0

strategy.entry("Short", strategy.short)

if bar\_index == last\_bar\_index - 10

strategy.close("Short")

// Calculate the amount of bars since the last closed trade.

barsSinceClosed = strategy.closedtrades > 0 ? bar\_index - strategy.closedtrades.exit\_bar\_index(strategy.closedtrades - 1) : na

plot(barsSinceClosed, "Bars since last closed trade")

EXAMPLE

// Calculates the average amount of bars per trade.

//**@version=**6

strategy("strategy.closedtrades.exit\_bar\_index Example 2")

// Enter long trades on three rising bars; exit on two falling bars.

if ta.rising(close, 3)

strategy.entry("Long", strategy.long)

if ta.falling(close, 2)

strategy.close("Long")

// Function that calculates the average amount of bars per trade.

avgBarsPerTrade() =>

sumBarsPerTrade = 0

for tradeNo = 0 to strategy.closedtrades - 1

// Loop through all closed trades, starting with the oldest.

sumBarsPerTrade += strategy.closedtrades.exit\_bar\_index(tradeNo) - strategy.closedtrades.entry\_bar\_index(tradeNo) + 1

result = nz(sumBarsPerTrade / strategy.closedtrades)

plot(avgBarsPerTrade())

SEE ALSO

[bar\_index](https://www.tradingview.com/pine-script-reference/v6/#var_bar_index)[last\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#var_last_bar_index)

**strategy.closedtrades.exit\_comment()**

Returns the comment message of the closed trade's exit, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) if there is no entry with this trade\_num.

SYNTAX

strategy.closedtrades.exit\_comment(trade\_num) → series string

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("`strategy.closedtrades.exit\_comment()` Example", overlay = true)

longCondition = ta.crossover(ta.sma(close, 14), ta.sma(close, 28))

if (longCondition)

strategy.entry("Long", strategy.long)

strategy.exit("Exit", stop = open \* 0.95, limit = close \* 1.05, trail\_points = 100, trail\_offset = 0, comment\_profit = "TP", comment\_loss = "SL", comment\_trailing = "TRAIL")

exitStats() =>

**int** slCount = 0

**int** tpCount = 0

**int** trailCount = 0

if strategy.closedtrades > 0

for i = 0 to strategy.closedtrades - 1

switch strategy.closedtrades.exit\_comment(i)

"TP" => tpCount += 1

"SL" => slCount += 1

"TRAIL" => trailCount += 1

[slCount, tpCount, trailCount]

var testTable = table.new(position.top\_right, 1, 4, color.orange, border\_width = 1)

if barstate.islastconfirmedhistory

[slCount, tpCount, trailCount] = exitStats()

table.cell(testTable, 0, 0, "Closed trades (" + str.tostring(strategy.closedtrades) +") stats:")

table.cell(testTable, 0, 1, "Stop Loss: " + str.tostring(slCount))

table.cell(testTable, 0, 2, "Take Profit: " + str.tostring(tpCount))

table.cell(testTable, 0, 3, "Trailing Stop: " + str.tostring(trailCount))

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)[strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit)[strategy.close](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.close)[strategy.closedtrades](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades)

**strategy.closedtrades.exit\_id()**

Returns the id of the closed trade's exit.

SYNTAX

strategy.closedtrades.exit\_id(trade\_num) → series string

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.closedtrades.exit\_id Example", overlay = true)

// Strategy calls to create single short and long trades

if bar\_index == last\_bar\_index - 15

strategy.entry("Long Entry", strategy.long)

else if bar\_index == last\_bar\_index - 10

strategy.entry("Short Entry", strategy.short)

// When a new open trade is detected then we create the exit strategy corresponding with the matching entry id

// We detect the correct entry id by determining if a position is long or short based on the position quantity

if ta.change(strategy.opentrades) != 0

posSign = strategy.opentrades.size(strategy.opentrades - 1)

strategy.exit(posSign > 0 ? "SL Long Exit" : "SL Short Exit", strategy.opentrades.entry\_id(strategy.opentrades - 1), stop = posSign > 0 ? high - ta.tr : low + ta.tr)

// When a new closed trade is detected then we place a label above the bar with the exit info

if ta.change(strategy.closedtrades) != 0

msg = "Trade closed by: " + strategy.closedtrades.exit\_id(strategy.closedtrades - 1)

label.new(bar\_index, high + (3 \* ta.tr), msg)

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

[strategy.closedtrades.exit\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.exit_bar_index)[strategy.closedtrades.exit\_price](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.exit_price)[strategy.closedtrades.exit\_time](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.exit_time)

**strategy.closedtrades.exit\_price()**

Returns the price of the closed trade's exit.

SYNTAX

strategy.closedtrades.exit\_price(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.closedtrades.exit\_price Example 1")

// We are creating a long trade every 5 bars

if bar\_index % 5 == 0

strategy.entry("Long", strategy.long)

strategy.close("Long")

// Return the exit price from the latest closed trade.

exitPrice = strategy.closedtrades.exit\_price(strategy.closedtrades - 1)

plot(exitPrice, "Long exit price")

EXAMPLE

// Calculates the average profit percentage for all closed trades.

//**@version=**6

strategy("strategy.closedtrades.exit\_price Example 2")

// Strategy calls to create single short and long trades.

if bar\_index == last\_bar\_index - 15

strategy.entry("Long Entry", strategy.long)

else if bar\_index == last\_bar\_index - 10

strategy.close("Long Entry")

strategy.entry("Short", strategy.short)

else if bar\_index == last\_bar\_index - 5

strategy.close("Short")

// Calculate profit for both closed trades.

profitPct = 0.0

for tradeNo = 0 to strategy.closedtrades - 1

entryP = strategy.closedtrades.entry\_price(tradeNo)

exitP = strategy.closedtrades.exit\_price(tradeNo)

profitPct += (exitP - entryP) / entryP \* strategy.closedtrades.size(tradeNo) \* 100

// Calculate average profit percent for both closed trades.

avgProfitPct = nz(profitPct / strategy.closedtrades)

plot(avgProfitPct)

SEE ALSO

[strategy.closedtrades.entry\_price](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.entry_price)

**strategy.closedtrades.exit\_time()**

Returns the UNIX time of the closed trade's exit, expressed in milliseconds.

SYNTAX

strategy.closedtrades.exit\_time(trade\_num) → series int

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.closedtrades.exit\_time Example 1")

// Enter long trades on three rising bars; exit on two falling bars.

if ta.rising(close, 3)

strategy.entry("Long", strategy.long)

if ta.falling(close, 2)

strategy.close("Long")

// Calculate the average trade duration.

avgTradeDuration() =>

sumTradeDuration = 0

for i = 0 to strategy.closedtrades - 1

sumTradeDuration += strategy.closedtrades.exit\_time(i) - strategy.closedtrades.entry\_time(i)

result = nz(sumTradeDuration / strategy.closedtrades)

// Display average duration converted to seconds and formatted using 2 decimal points.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, str.tostring(avgTradeDuration() / 1000, "#.##") + " seconds")

EXAMPLE

// Reopens a closed trade after X seconds.

//**@version=**6

strategy("strategy.closedtrades.exit\_time Example 2")

// Strategy calls to emulate a single long trade at the first bar.

if bar\_index == 0

strategy.entry("Long", strategy.long)

reopenPositionAfter(timeSec) =>

if strategy.closedtrades > 0

if time - strategy.closedtrades.exit\_time(strategy.closedtrades - 1) >= timeSec \* 1000

strategy.entry("Long", strategy.long)

// Reopen last closed position after 120 sec.

reopenPositionAfter(120)

if ta.change(strategy.opentrades) != 0

strategy.exit("Long", stop = low \* 0.9, profit = high \* 2.5)

SEE ALSO

[strategy.closedtrades.entry\_time](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency).

SYNTAX

strategy.closedtrades.max\_drawdown(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("`strategy.closedtrades.max\_drawdown` Example")

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Get the biggest max trade drawdown value from all of the closed trades.

maxTradeDrawDown() =>

maxDrawdown = 0.0

for tradeNo = 0 to strategy.closedtrades - 1

maxDrawdown := math.max(maxDrawdown, strategy.closedtrades.max\_drawdown(tradeNo))

result = maxDrawdown

plot(maxTradeDrawDown(), "Biggest max drawdown")

REMARKS

The function returns na if trade\_num is not in the range: 0 to strategy.closedtrades - 1.

SEE ALSO

[strategy.opentrades.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.max_drawdown)[strategy.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#var_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

strategy.closedtrades.max\_drawdown\_percent(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

SEE ALSO

[strategy.closedtrades.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.max_drawdown)[strategy.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency).

SYNTAX

strategy.closedtrades.max\_runup(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("`strategy.closedtrades.max\_runup` Example")

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Get the biggest max trade runup value from all of the closed trades.

maxTradeRunUp() =>

maxRunup = 0.0

for tradeNo = 0 to strategy.closedtrades - 1

maxRunup := math.max(maxRunup, strategy.closedtrades.max\_runup(tradeNo))

result = maxRunup

plot(maxTradeRunUp(), "Max trade runup")

SEE ALSO

[strategy.opentrades.max\_runup](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.max_runup)[strategy.max\_runup](https://www.tradingview.com/pine-script-reference/v6/#var_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

strategy.closedtrades.max\_runup\_percent(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

SEE ALSO

[strategy.closedtrades.max\_runup](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.max_runup)[strategy.max\_runup](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_runup)

**strategy.closedtrades.profit()**

Returns the profit/loss of the closed trade, expressed in [strategy.account\_currency](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency). Losses are expressed as negative values.

SYNTAX

strategy.closedtrades.profit(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("`strategy.closedtrades.profit` Example")

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Calculate average gross profit by adding the difference between gross profit and commission.

avgGrossProfit() =>

sumGrossProfit = 0.0

for tradeNo = 0 to strategy.closedtrades - 1

sumGrossProfit += strategy.closedtrades.profit(tradeNo) - strategy.closedtrades.commission(tradeNo)

result = nz(sumGrossProfit / strategy.closedtrades)

plot(avgGrossProfit(), "Average gross profit")

SEE ALSO

[strategy.opentrades.profit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.profit)[strategy.closedtrades.commission](https://www.tradingview.com/pine-script-reference/v6/#fun_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

strategy.closedtrades.profit\_percent(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

SEE ALSO

[strategy.closedtrades.profit](https://www.tradingview.com/pine-script-reference/v6/#fun_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

strategy.closedtrades.size(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("`strategy.closedtrades.size` Example 1")

// We calculate the max amt of shares we can buy.

amtShares = math.floor(strategy.equity / close)

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long, qty = amtShares)

if bar\_index % 20 == 0

strategy.close("Long")

// Plot the number of contracts traded in the last closed trade.

plot(strategy.closedtrades.size(strategy.closedtrades - 1), "Number of contracts traded")

EXAMPLE

// Calculates the average profit percentage for all closed trades.

//**@version=**6

strategy("`strategy.closedtrades.size` Example 2")

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Calculate profit for both closed trades.

profitPct = 0.0

for tradeNo = 0 to strategy.closedtrades - 1

entryP = strategy.closedtrades.entry\_price(tradeNo)

exitP = strategy.closedtrades.exit\_price(tradeNo)

profitPct += (exitP - entryP) / entryP \* strategy.closedtrades.size(tradeNo) \* 100

// Calculate average profit percent for both closed trades.

avgProfitPct = nz(profitPct / strategy.closedtrades)

plot(avgProfitPct)

SEE ALSO

[strategy.opentrades.size](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.size)[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.closedtrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.closedtrades)[strategy.opentrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.opentrades)

**strategy.convert\_to\_account()**

Converts the value from the currency that the symbol on the chart is traded in ([syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency)) to the currency used by the strategy ([strategy.account\_currency](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency)).

SYNTAX

strategy.convert\_to\_account(value) → series float

ARGUMENTS

**value (series int/float)** The value to be converted.

EXAMPLE

//**@version=**6

strategy("`strategy.convert\_to\_account` Example 1", currency = currency.EUR)

plot(close, "Close price using default currency")

plot(strategy.convert\_to\_account(close), "Close price converted to strategy currency")

EXAMPLE

// Calculates the "Buy and hold return" using your account's currency.

//**@version=**6

strategy("`strategy.convert\_to\_account` Example 2", currency = currency.EUR)

dateInput = input.time(timestamp("20 Jul 2021 00:00 +0300"), "From Date", confirm = true)

buyAndHoldReturnPct(fromDate) =>

if time >= fromDate

money = close \* syminfo.pointvalue

var initialBal = strategy.convert\_to\_account(money)

(strategy.convert\_to\_account(money) - initialBal) / initialBal \* 100

plot(buyAndHoldReturnPct(dateInput))

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)[strategy.convert\_to\_symbol](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.convert_to_symbol)

**strategy.convert\_to\_symbol()**

Converts the value from the currency used by the strategy ([strategy.account\_currency](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency)) to the currency that the symbol on the chart is traded in ([syminfo.currency](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.currency)).

SYNTAX

strategy.convert\_to\_symbol(value) → series float

ARGUMENTS

**value (series int/float)** The value to be converted.

EXAMPLE

//**@version=**6

strategy("`strategy.convert\_to\_symbol` Example", currency = currency.EUR)

// Calculate the max qty we can buy using current chart's currency.

calcContracts(accountMoney) =>

math.floor(strategy.convert\_to\_symbol(accountMoney) / syminfo.pointvalue / close)

// Return max qty we can buy using 300 euros

qt = calcContracts(300)

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars using our custom qty.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long, qty = qt)

if bar\_index % 20 == 0

strategy.close("Long")

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)[strategy.convert\_to\_account](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.convert_to_account)

**strategy.default\_entry\_qty()**

Calculates the default quantity, in units, of an entry order from [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) or [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) function and their representation in the "Properties" tab of the strategy's settings.

SYNTAX

strategy.default\_entry\_qty(fill\_price) → series float

ARGUMENTS

**fill\_price (series int/float)** The fill price for which to calculate the default order quantity.

EXAMPLE

//**@version=**6

strategy("Supertrend Strategy", overlay = true, default\_qty\_type = strategy.percent\_of\_equity, default\_qty\_value = 15)

//**@variable** The length of the ATR calculation.

atrPeriod = input(10, "ATR Length")

//**@variable** The ATR multiplier.

factor = input.float(3.0, "Factor", step = 0.01)

//**@variable** The tick offset of the stop order.

stopOffsetInput = input.int(100, "Tick offset for entry stop")

// Get the direction of the SuperTrend.

[\_, direction] = ta.supertrend(factor, atrPeriod)

if ta.change(direction) < 0

//**@variable** The stop price of the entry order.

stopPrice = close + syminfo.mintick \* stopOffsetInput

//**@variable** The expected default fill quantity at the `stopPrice`. This value may not reflect actual qty of the filled order, because fill price may be different.

calculatedQty = strategy.default\_entry\_qty(stopPrice)

strategy.entry("My Long Entry Id", strategy.long, stop = stopPrice)

label.new(bar\_index, stopPrice, str.format("Stop set at {0}\nExpected qty at {0}: {1}", math.round\_to\_mintick(stopPrice), calculatedQty))

if ta.change(direction) > 0

strategy.close\_all()

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](https://www.tradingview.com/pine-script-reference/v6/#fun_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.

This value represents the default calculated quantity of an order.

Order placement commands can override the default value by explicitly passing a new qty value in the function call.

**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.

The resulting order's type depends on the limit and stop parameters. If the call does not contain limit or stop arguments, it creates a [market order](https://www.tradingview.com/pine-script-docs/concepts/strategies/#market-orders) that executes on the next tick. If the call specifies a limit value but no stop value, it places a [limit order](https://www.tradingview.com/pine-script-docs/concepts/strategies/#limit-orders) that executes after the market price reaches the limit value or a better price (lower for buy orders and higher for sell orders). If the call specifies a stop value but no limit value, it places a [stop order](https://www.tradingview.com/pine-script-docs/concepts/strategies/#stop-and-stop-limit-orders) that executes after the market price reaches the stop value or a worse price (higher for buy orders and lower for sell orders). If the call contains limit and stop arguments, it creates a [stop-limit](https://www.tradingview.com/pine-script-docs/concepts/strategies/#stop-and-stop-limit-orders) order, which generates a limit order at the limit price only after the market price reaches the stop value or a worse price.

Orders from this command, unlike those from [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order), are affected by the pyramiding parameter of the [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) declaration statement. Pyramiding specifies the number of concurrent open entries allowed per position. For example, with pyramiding = 3, the strategy can have up to three open trades, and the command cannot create orders to open additional trades until at least one existing trade closes.

By default, when a strategy executes an order from this command in the opposite direction of the current market position, it reverses that position. For example, if there is an open long position of five shares, an order from this command with a qty of 5 and a direction of [strategy.short](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.short) triggers the sale of 10 shares to close the long position and open a new five-share short position. Users can change this behavior by specifying an allowed direction with the [strategy.risk\_allow\_entry\_in](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.risk_allow_entry_in) function.

SYNTAX

strategy.entry(id, direction, qty, limit, stop, oca\_name, oca\_type, comment, alert\_message, disable\_alert) → void

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](https://www.tradingview.com/pine-script-reference/v6/#var_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.

**direction (series strategy\_direction)** The direction of the trade. Possible values: [strategy.long](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.long) for a long trade, [strategy.short](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.short) for a short one.

**qty (series int/float)** Optional. The number of contracts/shares/lots/units in the resulting open trade when the order fills. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), which means that the command uses the default\_qty\_type and default\_qty\_value parameters of the [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) declaration statement to determine the quantity.

**limit (series int/float)** Optional. The limit price of the order. If specified, the command creates a limit or stop-limit order, depending on whether the stop value is also specified. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), which means the resulting order is not of the limit or stop-limit type.

**stop (series int/float)** Optional. The stop price of the order. If specified, the command creates a stop or stop-limit order, depending on whether the limit value is also specified. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), which means the resulting order is not of the stop or stop-limit type.

**oca\_name (series string)** Optional. The name of the order's One-Cancels-All (OCA) group. When a pending order with the same oca\_name and oca\_type parameters executes, that order affects all unfilled orders in the group. The default is an empty string, which means the order does not belong to an OCA group.

**oca\_type (input string)** Optional. Specifies how an unfilled order behaves when another pending order with the same oca\_name and oca\_type values executes. Possible values: [strategy.oca.cancel](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.oca.cancel), [strategy.oca.reduce](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.oca.reduce), [strategy.oca.none](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.oca.none). The default is [strategy.oca.none](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.oca.none).

**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 specified id. The default is an empty string.

**alert\_message (series string)** Optional. Custom text for the alert that fires when an order fills. If the "Message" field of the "Create Alert" dialog box contains the {{strategy.order.alert\_message}} placeholder, the alert message replaces the placeholder with this text. The default is an empty string.

**disable\_alert (series bool)** Optional. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) when the command creates an order, the strategy does not trigger an alert when that order fills. This parameter accepts a "series" value, meaning users can control which orders trigger alerts when they execute. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

strategy("Market order strategy", overlay = true)

// Calculate a 14-bar and 28-bar moving average of `close` prices.

**float** sma14 = ta.sma(close, 14)

**float** sma28 = ta.sma(close, 28)

// Place a market order to close the short trade and enter a long position when `sma14` crosses over `sma28`.

if ta.crossover(sma14, sma28)

strategy.entry("My Long Entry ID", strategy.long)

// Place a market order to close the long trade and enter a short position when `sma14` crosses under `sma28`.

if ta.crossunder(sma14, sma28)

strategy.entry("My Short Entry ID", strategy.short)

EXAMPLE

//**@version=**6

strategy("Limit order strategy", overlay=true, margin\_long=100, margin\_short=100)

//**@variable** The distance from the `close` price for each limit order.

**float** limitOffsetInput = input.int(100, "Limit offset, in ticks", 1) \* syminfo.mintick

//**@function** Draws a label and line at the specified `price` to visualize a limit order's level.

drawLimit(**float** price, **bool** isLong) =>

**color** col = isLong ? color.blue : color.red

label.new(

bar\_index, price, (isLong ? "Long" : "Short") + " limit order created",

style = label.style\_label\_right, color = col, textcolor = color.white

)

line.new(bar\_index, price, bar\_index + 1, price, extend = extend.right, style = line.style\_dashed, color = col)

//**@function** Stops the `l` line from extending further.

method stopExtend(**line** l) =>

l.set\_x2(bar\_index)

l.set\_extend(extend.none)

// Initialize two `line` variables to reference limit line IDs.

var **line** longLimit = na

var **line** shortLimit = na

// Calculate a 14-bar and 28-bar moving average of `close` prices.

**float** sma14 = ta.sma(close, 14)

**float** sma28 = ta.sma(close, 28)

if ta.crossover(sma14, sma28)

// Cancel any unfilled sell orders with the specified ID.

strategy.cancel("My Short Entry ID")

//**@variable** The limit price level. Its value is `limitOffsetInput` ticks below the current `close`.

**float** limitLevel = close - limitOffsetInput

// Place a long limit order to close the short trade and enter a long position at the `limitLevel`.

strategy.entry("My Long Entry ID", strategy.long, limit = limitLevel)

// Make new drawings for the long limit and stop extending the `shortLimit` line.

longLimit := drawLimit(limitLevel, isLong = true)

shortLimit.stopExtend()

if ta.crossunder(sma14, sma28)

// Cancel any unfilled buy orders with the specified ID.

strategy.cancel("My Long Entry ID")

//**@variable** The limit price level. Its value is `limitOffsetInput` ticks above the current `close`.

**float** limitLevel = close + limitOffsetInput

// Place a short limit order to close the long trade and enter a short position at the `limitLevel`.

strategy.entry("My Short Entry ID", strategy.short, limit = limitLevel)

// Make new drawings for the short limit and stop extending the `shortLimit` line.

shortLimit := drawLimit(limitLevel, isLong = false)

longLimit.stopExtend()

**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](https://www.tradingview.com/pine-script-docs/concepts/strategies/#market-orders). To exit from a position with a market order, use [strategy.close](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.close) or [strategy.close\_all](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.close_all).

If a call to this command contains a profit or limit argument, it creates [take-profit](https://www.tradingview.com/pine-script-docs/concepts/strategies/#take-profit-and-stop-loss) orders to exit from applicable trades at the determined price levels or better values (higher for long trades and lower for short ones). If the call contains loss or stop arguments, it creates [stop-loss](https://www.tradingview.com/pine-script-docs/concepts/strategies/#take-profit-and-stop-loss) orders to exit from applicable trades at the determined levels or worse values (lower for long trades and higher for short ones). Calling this command with profit or limit and loss or stop arguments creates an order bracket with both order types.

This command can create [trailing stop](https://www.tradingview.com/pine-script-docs/concepts/strategies/#trailing-stops) orders when its call specifies a trail\_price or trail\_points argument and a trail\_offset argument. A trailing stop order activates when the price moves trail\_points ticks past the entry price or touches the trail\_price level. Once activated, the stop follows trail\_offset ticks behind the market price each time the trade's profit reaches a new high. The stop does not move when the trade does not achieve a new best value.

Each call to this command reserves a portion of the position to close until the strategy fills or cancels its orders. For example, if there is an open position of 50 contracts and a [strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit) call specifies a qty of 20, that call's orders reserve 20 contracts out of the position. A second call can close a maximum of 30 contracts, even if its qty is 50 and one of its orders executes first. This behavior does not affect the orders from other commands, such as [strategy.close](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.close) or [strategy.order](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.order).

If a call to this command occurs before a created entry order's execution, the strategy waits and does not create the exit orders until after the entry order executes.

SYNTAX

strategy.exit(id, from\_entry, qty, qty\_percent, profit, limit, loss, stop, trail\_price, trail\_points, trail\_offset, oca\_name, comment, comment\_profit, comment\_loss, comment\_trailing, alert\_message, alert\_profit, alert\_loss, alert\_trailing, disable\_alert) → void

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.

**from\_entry (series string)** Optional. The entry order ID of the trade to exit from. If there is more than one open trade with the specified entry ID, the command generates exit orders for all the entries from before or at the time of the call. The default is an empty string, which means the command generates exit orders for all open trades until the position closes.

**qty (series int/float)** Optional. The number of contracts/lots/shares/units to close when an exit order fills. If specified, the command uses this value instead of qty\_percent to determine the order size. The exit orders reserve this quantity from the position, meaning other calls to this command cannot close this portion until the strategy fills or cancels those orders. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), which means the order size depends on the qty\_percent value.

**qty\_percent (series int/float)** Optional. A value between 0 and 100 representing the percentage of the open trade quantity to close when an exit order fills. The exit orders reserve this percentage from the applicable open trades, meaning other calls to this command cannot close this portion until the strategy fills or cancels those orders. The percentage calculation depends on the total size of the applicable open trades without considering the reserved amount from other [strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit) calls. The command ignores this parameter if the qty value is not [na](https://www.tradingview.com/pine-script-reference/v6/#var_na). The default is 100.

**profit (series int/float)** Optional. The take-profit distance, expressed in ticks. If specified, the command creates a limit order to exit the trade profit ticks away from the entry price in the favorable direction. The order executes at the calculated price or a better value. If this parameter and limit are not [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), the command places a take-profit order only at the price level expected to trigger an exit first. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**limit (series int/float)** Optional. The take-profit price. If this parameter and profit are not [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), the command places a take-profit order only at the price level expected to trigger an exit first. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**loss (series int/float)** Optional. The stop-loss distance, expressed in ticks. If specified, the command creates a stop order to exit the trade loss ticks away from the entry price in the unfavorable direction. The order executes at the calculated price or a worse value. If this parameter and stop are not [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), the command places a stop-loss order only at the price level expected to trigger an exit first. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**stop (series int/float)** Optional. The stop-loss price. If this parameter and loss are not [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), the command places a stop-loss order only at the price level expected to trigger an exit first. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**trail\_price (series int/float)** Optional. The price of the trailing stop activation level. If the value is more favorable than the entry price, the command creates a trailing stop when the market price reaches that value. If less favorable than the entry price, the command creates the trailing stop immediately when the current market price is equal to or more favorable than the value. If this parameter and trail\_points are not [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), the command sets the activation level using the value expected to activate the stop first. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**trail\_points (series int/float)** Optional. The trailing stop activation distance, expressed in ticks. If the value is positive, the command creates a trailing stop order when the market price moves trail\_points ticks away from the trade's entry price in the favorable direction. If the value is negative, the command creates the trailing stop immediately when the market price is equal to or more favorable than the level trail\_points ticks away from the entry price in the unfavorable direction. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**trail\_offset (series int/float)** Optional. The trailing stop offset. When the market price reaches the activation level determined by the trail\_price or trail\_points parameter, or exceeds the level in the favorable direction, the command creates a trailing stop with an initial value trail\_offset ticks away from that level in the unfavorable direction. After activation, the trailing stop moves toward the market price each time the trade's profit reaches a better value, maintaining the specified distance behind the best price. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

**oca\_name (series string)** Optional. The name of the One-Cancels-All (OCA) group that the command's take-profit, stop-loss, and trailing stop orders belong to. All orders from this command are of the [strategy.oca.reduce](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.oca.reduce) OCA type. When an order of this OCA type with the same oca\_name executes, the strategy reduces the sizes of other unfilled orders in the OCA group by the filled quantity. The default is an empty string, which means the strategy assigns the OCA name automatically, and the resulting orders cannot reduce or be reduced by the orders from other commands.

**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 specified id. The command ignores this value if the call includes an argument for a comment\_\* parameter that applies to the order. The default is an empty string.

**comment\_profit (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 specified id or comment. This comment applies only to the command's take-profit orders created using the profit or limit parameter. The default is an empty string.

**comment\_loss (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 specified id or comment. This comment applies only to the command's stop-loss orders created using the loss or stop parameter. The default is an empty string.

**comment\_trailing (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 specified id or comment. This comment applies only to the command's trailing stop orders created using the trail\_price or trail\_points and trail\_offset parameters. The default is an empty string.

**alert\_message (series string)** Optional. Custom text for the alert that fires when an order fills. If the "Message" field of the "Create Alert" dialog box contains the {{strategy.order.alert\_message}} placeholder, the alert message replaces the placeholder with this text. The command ignores this value if the call includes an argument for the other alert\_\* parameter that applies to the order. The default is an empty string.

**alert\_profit (series string)** Optional. Custom text for the alert that fires when an order fills. If the "Message" field of the "Create Alert" dialog box contains the {{strategy.order.alert\_message}} placeholder, the alert message replaces the placeholder with this text. This message applies only to the command's take-profit orders created using the profit or limit parameter. The default is an empty string.

**alert\_loss (series string)** Optional. Custom text for the alert that fires when an order fills. If the "Message" field of the "Create Alert" dialog box contains the {{strategy.order.alert\_message}} placeholder, the alert message replaces the placeholder with this text. This message applies only to the command's stop-loss orders created using the loss or stop parameter. The default is an empty string.

**alert\_trailing (series string)** Optional. Custom text for the alert that fires when an order fills. If the "Message" field of the "Create Alert" dialog box contains the {{strategy.order.alert\_message}} placeholder, the alert message replaces the placeholder with this text. This message applies only to the command's trailing stop orders created using the trail\_price or trail\_points and trail\_offset parameters. The default is an empty string.

**disable\_alert (series bool)** Optional. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) when the command creates an order, the strategy does not trigger an alert when that order fills. This parameter accepts a "series" value, meaning users can control which orders trigger alerts when they execute. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

strategy("Exit bracket strategy", overlay = true)

// Inputs that define the profit and loss amount of each trade as a tick distance from the entry price.

**int** profitDistanceInput = input.int(100, "Profit distance, in ticks", 1)

**int** lossDistanceInput = input.int(100, "Loss distance, in ticks", 1)

// Variables to track the take-profit and stop-loss price.

var **float** takeProfit = na

var **float** stopLoss = na

// Calculate a 14-bar and 28-bar moving average of `close` prices.

**float** sma14 = ta.sma(close, 14)

**float** sma28 = ta.sma(close, 28)

if ta.crossover(sma14, sma28) and strategy.opentrades == 0

// Place a market order to enter a long position.

strategy.entry("My Long Entry ID", strategy.long)

// Place a take-profit and stop-loss order when the entry order fills.

strategy.exit("My Long Exit ID", "My Long Entry ID", profit = profitDistanceInput, loss = lossDistanceInput)

if ta.change(strategy.opentrades) == 1

//**@variable** The long entry price.

**float** entryPrice = strategy.opentrades.entry\_price(0)

// Update the `takeProfit` and `stopLoss` values.

takeProfit := entryPrice + profitDistanceInput \* syminfo.mintick

stopLoss := entryPrice - lossDistanceInput \* syminfo.mintick

if ta.change(strategy.closedtrades) == 1

// Reset the `takeProfit` and `stopLoss`.

takeProfit := na

stopLoss := na

// Plot the `takeProfit` and `stopLoss`.

plot(takeProfit, "Take-profit level", color.green, 2, plot.style\_linebr)

plot(stopLoss, "Stop-loss level", color.red, 2, plot.style\_linebr)

EXAMPLE

//**@version=**6

strategy("Trailing stop strategy", overlay = true)

//**@variable** The distance required to activate the trailing stop.

**float** activationDistanceInput = input.int(100, "Trail activation distance, in ticks") \* syminfo.mintick

//**@variable** The number of ticks the trailing stop follows behind the price as it reaches new peaks.

**int** trailDistanceInput = input.int(100, "Trail distance, in ticks")

//**@function** Draws a label and line at the specified `price` to visualize a trailing stop order's activation level.

drawActivation(**float** price) =>

label.new(

bar\_index, price, "Activation level", style = label.style\_label\_right,

color = color.gray, textcolor = color.white

)

line.new(

bar\_index, price, bar\_index + 1, price, extend = extend.right, style = line.style\_dashed, color = color.gray

)

//**@function** Stops the `l` line from extending further.

method stopExtend(**line** l) =>

l.set\_x2(bar\_index)

l.set\_extend(extend.none)

// The activation line, active trailing stop price, and active trailing stop flag.

var **line** activationLine = na

var **float** trailingStopPrice = na

var **bool** isActive = false

if bar\_index % 100 == 0 and strategy.opentrades == 0

trailingStopPrice := na

isActive := false

// Place a market order to enter a long position.

strategy.entry("My Long Entry ID", strategy.long)

//**@variable** The activation level's price.

**float** activationPrice = close + activationDistanceInput

// Create a trailing stop order that activates the defined number of ticks above the entry price.

strategy.exit(

"My Long Exit ID", "My Long Entry ID", trail\_price = activationPrice, trail\_offset = trailDistanceInput,

oca\_name = "Exit"

)

// Create new drawings at the `activationPrice`.

activationLine := drawActivation(activationPrice)

// Logic for trailing stop visualization.

if strategy.opentrades == 1

// Stop extending the `activationLine` when the stop activates.

if not isActive and high > activationLine.get\_price(bar\_index)

isActive := true

activationLine.stopExtend()

// Update the `trailingStopPrice` while the trailing stop is active.

if isActive

**float** offsetPrice = high - trailDistanceInput \* syminfo.mintick

trailingStopPrice := math.max(nz(trailingStopPrice, offsetPrice), offsetPrice)

// Close the trade with a market order if the trailing stop does not activate before the next 300th bar.

if not isActive and bar\_index % 300 == 0

strategy.close\_all("Market close")

// Reset the `trailingStopPrice` and `isActive` flags when the trade closes, and stop extending the `activationLine`.

if ta.change(strategy.closedtrades) > 0

if not isActive

activationLine.stopExtend()

trailingStopPrice := na

isActive := false

// Plot the `trailingStopPrice`.

plot(trailingStopPrice, "Trailing stop", color.red, 3, plot.style\_linebr)

REMARKS

A single call to the [strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-docs/concepts/strategies/#exits-for-multiple-entries) section of our User Manual to learn more.

When a position consists of several open trades, and the close\_entries\_rule in the [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) declaration statement is "FIFO" (default), the orders from a [strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit) call exit from the position starting with the first open trade. This behavior applies even if the from\_entry value is the entry ID of different open trades. However, in that case, the maximum size of the exit orders still depends on the trades opened by orders with the from\_entry ID. For more information, see [this](https://www.tradingview.com/pine-script-docs/concepts/strategies/#closing-a-market-position) section of our User Manual.

If a [strategy.exit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.exit) call includes arguments for creating stop-loss and trailing stop orders, the command places only the order that is supposed to fill first, because both orders are of the "stop" type.

**strategy.opentrades.commission()**

Returns the sum of entry and exit fees paid in the open trade, expressed in [strategy.account\_currency](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency).

SYNTAX

strategy.opentrades.commission(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the open trade. The number of the first trade is zero.

EXAMPLE

// Calculates the gross profit or loss for the current open position.

//**@version=**6

strategy("`strategy.opentrades.commission` Example", commission\_type = strategy.commission.percent, commission\_value = 0.1)

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Calculate gross profit or loss for open positions only.

tradeOpenGrossPL() =>

sumOpenGrossPL = 0.0

for tradeNo = 0 to strategy.opentrades - 1

sumOpenGrossPL += strategy.opentrades.profit(tradeNo) - strategy.opentrades.commission(tradeNo)

result = sumOpenGrossPL

plot(tradeOpenGrossPL())

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)[strategy.closedtrades.commission](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.commission)

**strategy.opentrades.entry\_bar\_index()**

Returns the bar\_index of the open trade's entry.

SYNTAX

strategy.opentrades.entry\_bar\_index(trade\_num) → series int

ARGUMENTS

**trade\_num (series int)** The trade number of the open trade. The number of the first trade is zero.

EXAMPLE

// Wait 10 bars and then close the position.

//**@version=**6

strategy("`strategy.opentrades.entry\_bar\_index` Example")

barsSinceLastEntry() =>

strategy.opentrades > 0 ? bar\_index - strategy.opentrades.entry\_bar\_index(strategy.opentrades - 1) : na

// Enter a long position if there are no open positions.

if strategy.opentrades == 0

strategy.entry("Long", strategy.long)

// Close the long position after 10 bars.

if barsSinceLastEntry() >= 10

strategy.close("Long")

SEE ALSO

[strategy.closedtrades.entry\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.entry_bar_index)[strategy.closedtrades.exit\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.exit_bar_index)

**strategy.opentrades.entry\_comment()**

Returns the comment message of the open trade's entry, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_na) if there is no entry with this trade\_num.

SYNTAX

strategy.opentrades.entry\_comment(trade\_num) → series string

ARGUMENTS

**trade\_num (series int)** The trade number of the open trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("`strategy.opentrades.entry\_comment()` Example", overlay = true)

stopPrice = open \* 1.01

longCondition = ta.crossover(ta.sma(close, 14), ta.sma(close, 28))

if (longCondition)

strategy.entry("Long", strategy.long, stop = stopPrice, comment = str.tostring(stopPrice, "#.####"))

var testTable = table.new(position.top\_right, 1, 3, color.orange, border\_width = 1)

if barstate.islastconfirmedhistory or barstate.isrealtime

table.cell(testTable, 0, 0, 'Last entry stats')

table.cell(testTable, 0, 1, "Order stop price value: " + strategy.opentrades.entry\_comment(strategy.opentrades - 1))

table.cell(testTable, 0, 2, "Actual Entry Price: " + str.tostring(strategy.opentrades.entry\_price(strategy.opentrades - 1)))

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)[strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry)[strategy.opentrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.opentrades)

**strategy.opentrades.entry\_id()**

Returns the id of the open trade's entry.

SYNTAX

strategy.opentrades.entry\_id(trade\_num) → series string

ARGUMENTS

**trade\_num (series int)** The trade number of the open trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("`strategy.opentrades.entry\_id` Example", overlay = true)

// We enter a long position when 14 period sma crosses over 28 period sma.

// We enter a short position when 14 period sma crosses under 28 period sma.

longCondition = ta.crossover(ta.sma(close, 14), ta.sma(close, 28))

shortCondition = ta.crossunder(ta.sma(close, 14), ta.sma(close, 28))

// Strategy calls to enter a long or short position when the corresponding condition is met.

if longCondition

strategy.entry("Long entry at bar #" + str.tostring(bar\_index), strategy.long)

if shortCondition

strategy.entry("Short entry at bar #" + str.tostring(bar\_index), strategy.short)

// Display ID of the latest open position.

if barstate.islastconfirmedhistory

label.new(bar\_index, high + (2 \* ta.tr), "Last opened position is \n " + strategy.opentrades.entry\_id(strategy.opentrades - 1))

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

[strategy.opentrades.entry\_bar\_index](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.entry_bar_index)[strategy.opentrades.entry\_price](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.entry_price)[strategy.opentrades.entry\_time](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.entry_time)

**strategy.opentrades.entry\_price()**

Returns the price of the open trade's entry.

SYNTAX

strategy.opentrades.entry\_price(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the open trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.opentrades.entry\_price Example 1", overlay = true)

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if ta.crossover(close, ta.sma(close, 14))

strategy.entry("Long", strategy.long)

// Return the entry price for the latest closed trade.

currEntryPrice = strategy.opentrades.entry\_price(strategy.opentrades - 1)

currExitPrice = currEntryPrice \* 1.05

if high >= currExitPrice

strategy.close("Long")

plot(currEntryPrice, "Long entry price", style = plot.style\_linebr)

plot(currExitPrice, "Long exit price", color.green, style = plot.style\_linebr)

EXAMPLE

// Calculates the average price for the open position.

//**@version=**6

strategy("strategy.opentrades.entry\_price Example 2", pyramiding = 2)

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Calculates the average price for the open position.

avgOpenPositionPrice() =>

sumOpenPositionPrice = 0.0

for tradeNo = 0 to strategy.opentrades - 1

sumOpenPositionPrice += strategy.opentrades.entry\_price(tradeNo) \* strategy.opentrades.size(tradeNo) / strategy.position\_size

result = nz(sumOpenPositionPrice / strategy.opentrades)

plot(avgOpenPositionPrice())

SEE ALSO

[strategy.closedtrades.exit\_price](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.exit_price)

**strategy.opentrades.entry\_time()**

Returns the UNIX time of the open trade's entry, expressed in milliseconds.

SYNTAX

strategy.opentrades.entry\_time(trade\_num) → series int

ARGUMENTS

**trade\_num (series int)** The trade number of the open trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.opentrades.entry\_time Example")

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Calculates duration in milliseconds since the last position was opened.

timeSinceLastEntry()=>

strategy.opentrades > 0 ? (time - strategy.opentrades.entry\_time(strategy.opentrades - 1)) : na

plot(timeSinceLastEntry() / 1000 \* 60 \* 60 \* 24, "Days since last entry")

SEE ALSO

[strategy.closedtrades.entry\_time](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.entry_time)[strategy.closedtrades.exit\_time](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency).

SYNTAX

strategy.opentrades.max\_drawdown(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the open trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.opentrades.max\_drawdown Example 1")

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Plot the max drawdown of the latest open trade.

plot(strategy.opentrades.max\_drawdown(strategy.opentrades - 1), "Max drawdown of the latest open trade")

EXAMPLE

// Calculates the max trade drawdown value for all open trades.

//**@version=**6

strategy("`strategy.opentrades.max\_drawdown` Example 2", pyramiding = 100)

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Get the biggest max trade drawdown value from all of the open trades.

maxTradeDrawDown() =>

maxDrawdown = 0.0

for tradeNo = 0 to strategy.opentrades - 1

maxDrawdown := math.max(maxDrawdown, strategy.opentrades.max\_drawdown(tradeNo))

result = maxDrawdown

plot(maxTradeDrawDown(), "Biggest max drawdown")

REMARKS

The function returns na if trade\_num is not in the range: 0 to strategy.closedtrades - 1.

SEE ALSO

[strategy.closedtrades.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.max_drawdown)[strategy.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#var_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

strategy.opentrades.max\_drawdown\_percent(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

SEE ALSO

[strategy.opentrades.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.max_drawdown)[strategy.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency).

SYNTAX

strategy.opentrades.max\_runup(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the open trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("strategy.opentrades.max\_runup Example 1")

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Plot the max runup of the latest open trade.

plot(strategy.opentrades.max\_runup(strategy.opentrades - 1), "Max runup of the latest open trade")

EXAMPLE

// Calculates the max trade runup value for all open trades.

//**@version=**6

strategy("strategy.opentrades.max\_runup Example 2", pyramiding = 100)

// Enter a long position every 30 bars.

if bar\_index % 30 == 0

strategy.entry("Long", strategy.long)

// Calculate biggest max trade runup value from all of the open trades.

maxOpenTradeRunUp() =>

maxRunup = 0.0

for tradeNo = 0 to strategy.opentrades - 1

maxRunup := math.max(maxRunup, strategy.opentrades.max\_runup(tradeNo))

result = maxRunup

plot(maxOpenTradeRunUp(), "Biggest max runup of all open trades")

SEE ALSO

[strategy.closedtrades.max\_runup](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.max_runup)[strategy.max\_drawdown](https://www.tradingview.com/pine-script-reference/v6/#var_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

strategy.opentrades.max\_runup\_percent(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

SEE ALSO

[strategy.opentrades.max\_runup](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.opentrades.max_runup)[strategy.max\_runup](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.max_runup)

**strategy.opentrades.profit()**

Returns the profit/loss of the open trade, expressed in [strategy.account\_currency](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.account_currency). Losses are expressed as negative values.

SYNTAX

strategy.opentrades.profit(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the open trade. The number of the first trade is zero.

EXAMPLE

// Returns the profit of the last open trade.

//**@version=**6

strategy("`strategy.opentrades.profit` Example 1", commission\_type = strategy.commission.percent, commission\_value = 0.1)

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

plot(strategy.opentrades.profit(strategy.opentrades - 1), "Profit of the latest open trade")

EXAMPLE

// Calculates the profit for all open trades.

//**@version=**6

strategy("`strategy.opentrades.profit` Example 2", pyramiding = 5)

// Strategy calls to enter 5 long positions every 2 bars.

if bar\_index % 2 == 0

strategy.entry("Long", strategy.long, qty = 5)

// Calculate open profit or loss for the open positions.

tradeOpenPL() =>

sumProfit = 0.0

for tradeNo = 0 to strategy.opentrades - 1

sumProfit += strategy.opentrades.profit(tradeNo)

result = sumProfit

plot(tradeOpenPL(), "Profit of all open trades")

SEE ALSO

[strategy.closedtrades.profit](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.profit)[strategy.openprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.openprofit)[strategy.netprofit](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.netprofit)[strategy.grossprofit](https://www.tradingview.com/pine-script-reference/v6/#var_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

strategy.opentrades.profit\_percent(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the closed trade. The number of the first trade is zero.

SEE ALSO

[strategy.opentrades.profit](https://www.tradingview.com/pine-script-reference/v6/#fun_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

strategy.opentrades.size(trade\_num) → series float

ARGUMENTS

**trade\_num (series int)** The trade number of the open trade. The number of the first trade is zero.

EXAMPLE

//**@version=**6

strategy("`strategy.opentrades.size` Example 1")

// We calculate the max amt of shares we can buy.

amtShares = math.floor(strategy.equity / close)

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long, qty = amtShares)

if bar\_index % 20 == 0

strategy.close("Long")

// Plot the number of contracts in the latest open trade.

plot(strategy.opentrades.size(strategy.opentrades - 1), "Amount of contracts in latest open trade")

EXAMPLE

// Calculates the average profit percentage for all open trades.

//**@version=**6

strategy("`strategy.opentrades.size` Example 2")

// Strategy calls to enter long trades every 15 bars and exit long trades every 20 bars.

if bar\_index % 15 == 0

strategy.entry("Long", strategy.long)

if bar\_index % 20 == 0

strategy.close("Long")

// Calculate profit for all open trades.

profitPct = 0.0

for tradeNo = 0 to strategy.opentrades - 1

entryP = strategy.opentrades.entry\_price(tradeNo)

exitP = close

profitPct += (exitP - entryP) / entryP \* strategy.opentrades.size(tradeNo) \* 100

// Calculate average profit percent for all open trades.

avgProfitPct = nz(profitPct / strategy.opentrades)

plot(avgProfitPct)

SEE ALSO

[strategy.closedtrades.size](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.closedtrades.size)[strategy.position\_size](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.position_size)[strategy.opentrades](https://www.tradingview.com/pine-script-reference/v6/#var_strategy.opentrades)[strategy.closedtrades](https://www.tradingview.com/pine-script-reference/v6/#var_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.

The resulting order's type depends on the limit and stop parameters. If the call does not contain limit or stop arguments, it creates a [market order](https://www.tradingview.com/pine-script-docs/concepts/strategies/#market-orders) that executes on the next tick. If the call specifies a limit value but no stop value, it places a [limit order](https://www.tradingview.com/pine-script-docs/concepts/strategies/#limit-orders) that executes after the market price reaches the limit value or a better price (lower for buy orders and higher for sell orders). If the call specifies a stop value but no limit value, it places a [stop order](https://www.tradingview.com/pine-script-docs/concepts/strategies/#stop-and-stop-limit-orders) that executes after the market price reaches the stop value or a worse price (higher for buy orders and lower for sell orders). If the call contains limit and stop arguments, it creates a [stop-limit](https://www.tradingview.com/pine-script-docs/concepts/strategies/#stop-and-stop-limit-orders) order, which generates a limit order at the limit price only after the market price reaches the stop value or a worse price.

Orders from this command, unlike those from [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry), are not affected by the pyramiding parameter of the [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) declaration statement. Strategies can open any number of trades in the same direction with calls to this function.

This command does not automatically reverse open positions because it does not exclusively create entry orders like [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) does. For example, if there is an open long position of five shares, an order from this command with a qty of 5 and a direction of [strategy.short](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.short) triggers the sale of five shares, which closes the position.

SYNTAX

strategy.order(id, direction, qty, limit, stop, oca\_name, oca\_type, comment, alert\_message, disable\_alert) → void

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](https://www.tradingview.com/pine-script-reference/v6/#var_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.

**direction (series strategy\_direction)** The direction of the trade. Possible values: [strategy.long](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.long) for a long trade, [strategy.short](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.short) for a short one.

**qty (series int/float)** Optional. The number of contracts/shares/lots/units to trade when the order fills. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), which means that the command uses the default\_qty\_type and default\_qty\_value parameters of the [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) declaration statement to determine the quantity.

**limit (series int/float)** Optional. The limit price of the order. If specified, the command creates a limit or stop-limit order, depending on whether the stop value is also specified. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), which means the resulting order is not of the limit or stop-limit type.

**stop (series int/float)** Optional. The stop price of the order. If specified, the command creates a stop or stop-limit order, depending on whether the limit value is also specified. The default is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na), which means the resulting order is not of the stop or stop-limit type.

**oca\_name (series string)** Optional. The name of the order's One-Cancels-All (OCA) group. When a pending order with the same oca\_name and oca\_type parameters executes, that order affects all unfilled orders in the group. The default is an empty string, which means the order does not belong to an OCA group.

**oca\_type (input string)** Optional. Specifies how an unfilled order behaves when another pending order with the same oca\_name and oca\_type values executes. Possible values: [strategy.oca.cancel](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.oca.cancel), [strategy.oca.reduce](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.oca.reduce), [strategy.oca.none](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.oca.none). The default is [strategy.oca.none](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.oca.none).

**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 specified id. The default is an empty string.

**alert\_message (series string)** Optional. Custom text for the alert that fires when an order fills. If the "Message" field of the "Create Alert" dialog box contains the {{strategy.order.alert\_message}} placeholder, the alert message replaces the placeholder with this text. The default is an empty string.

**disable\_alert (series bool)** Optional. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) when the command creates an order, the strategy does not trigger an alert when that order fills. This parameter accepts a "series" value, meaning users can control which orders trigger alerts when they execute. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

strategy("Market order strategy", overlay = true)

// Calculate a 14-bar and 28-bar moving average of `close` prices.

**float** sma14 = ta.sma(close, 14)

**float** sma28 = ta.sma(close, 28)

// Place a market order to enter a long position when `sma14` crosses over `sma28`.

if ta.crossover(sma14, sma28) and strategy.position\_size == 0

strategy.order("My Long Entry ID", strategy.long)

// Place a market order to sell the same quantity as the long trade when `sma14` crosses under `sma28`,

// effectively closing the long position.

if ta.crossunder(sma14, sma28) and strategy.position\_size > 0

strategy.order("My Long Exit ID", strategy.short)

EXAMPLE

//**@version=**6

strategy("Limit and stop exit strategy", overlay = true)

//**@variable** The distance from the long entry price for each short limit order.

**float** shortOffsetInput = input.int(200, "Sell limit/stop offset, in ticks", 1) \* syminfo.mintick

//**@function** Draws a label and line at the specified `price` to visualize a limit order's level.

drawLimit(**float** price, **bool** isLong, **bool** isStop = false) =>

**color** col = isLong ? color.blue : color.red

label.new(

bar\_index, price, (isLong ? "Long " : "Short ") + (isStop ? "stop" : "limit") + " order created",

style = label.style\_label\_right, color = col, textcolor = color.white

)

line.new(bar\_index, price, bar\_index + 1, price, extend = extend.right, style = line.style\_dashed, color = col)

//**@function** Stops the `l` line from extending further.

method stopExtend(**line** l) =>

l.set\_x2(bar\_index)

l.set\_extend(extend.none)

// Initialize two `line` variables to reference limit and stop line IDs.

var **line** profitLimit = na

var **line** lossStop = na

// Calculate a 14-bar and 28-bar moving average of `close` prices.

**float** sma14 = ta.sma(close, 14)

**float** sma28 = ta.sma(close, 28)

if ta.crossover(sma14, sma28) and strategy.position\_size == 0

// Place a market order to enter a long position.

strategy.order("My Long Entry ID", strategy.long)

if strategy.position\_size > 0 and strategy.position\_size[1] == 0

//**@variable** The entry price of the long trade.

**float** entryPrice = strategy.opentrades.entry\_price(0)

// Calculate short limit and stop levels above and below the `entryPrice`.

**float** profitLevel = entryPrice + shortOffsetInput

**float** lossLevel = entryPrice - shortOffsetInput

// Place short limit and stop orders at the `profitLevel` and `lossLevel`.

strategy.order("Profit", strategy.short, limit = profitLevel, oca\_name = "Bracket", oca\_type = strategy.oca.cancel)

strategy.order("Loss", strategy.short, stop = lossLevel, oca\_name = "Bracket", oca\_type = strategy.oca.cancel)

// Make new drawings for the `profitLimit` and `lossStop` lines.

profitLimit := drawLimit(profitLevel, isLong = false)

lossStop := drawLimit(lossLevel, isLong = false, isStop = true)

if ta.change(strategy.closedtrades) > 0

// Stop extending the `profitLimit` and `lossStop` lines.

profitLimit.stopExtend()

lossStop.stopExtend()

**strategy.risk.allow\_entry\_in()**

This function can be used to specify in which market direction the [strategy.entry](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy.entry) function is allowed to open positions.

SYNTAX

strategy.risk.allow\_entry\_in(value) → void

ARGUMENTS

**value (simple string)** The allowed direction. Possible values: [strategy.direction.all](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.direction.all), [strategy.direction.long](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.direction.long), [strategy.direction.short](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.direction.short)

EXAMPLE

//**@version=**6

strategy("strategy.risk.allow\_entry\_in")

strategy.risk.allow\_entry\_in(strategy.direction.long)

if open > close

strategy.entry("Long", strategy.long)

// Instead of opening a short position with 10 contracts, this command will close long entries.

if open < close

strategy.entry("Short", strategy.short, qty = 10)

**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

strategy.risk.max\_cons\_loss\_days(count, alert\_message) → void

ARGUMENTS

**count (simple int)** A required parameter. The allowed number of consecutive days with losses.

**alert\_message (simple string)** An optional parameter which replaces the {{strategy.order.alert\_message}} placeholder when it is used in the "Create Alert" dialog box's "Message" field.

EXAMPLE

//**@version=**6

strategy("risk.max\_cons\_loss\_days Demo 1")

strategy.risk.max\_cons\_loss\_days(3) // No orders will be placed after 3 days, if each day is with loss.

plot(strategy.position\_size)

**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

strategy.risk.max\_drawdown(value, type, alert\_message) → void

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.

**type (simple string)** A required parameter. The type of the value. Please specify one of the following values: [strategy.percent\_of\_equity](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.percent_of_equity) or [strategy.cash](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.cash). Note: if equity drops down to zero or to a negative and the 'strategy.percent\_of\_equity' is specified, all pending orders are cancelled, all open positions are closed and no new orders can be placed for good.

**alert\_message (simple string)** An optional parameter which replaces the {{strategy.order.alert\_message}} placeholder when it is used in the "Create Alert" dialog box's "Message" field.

EXAMPLE

//**@version=**6

strategy("risk.max\_drawdown Demo 1")

strategy.risk.max\_drawdown(50, strategy.percent\_of\_equity) // set maximum drawdown to 50% of maximum equity

plot(strategy.position\_size)

EXAMPLE

//**@version=**6

strategy("risk.max\_drawdown Demo 2", currency = "EUR")

strategy.risk.max\_drawdown(2000, strategy.cash) // set maximum drawdown to 2000 EUR from maximum equity

plot(strategy.position\_size)

**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

strategy.risk.max\_intraday\_filled\_orders(count, alert\_message) → void

ARGUMENTS

**count (simple int)** A required parameter. The maximum number of filled orders per 1 day.

**alert\_message (simple string)** An optional parameter which replaces the {{strategy.order.alert\_message}} placeholder when it is used in the "Create Alert" dialog box's "Message" field.

EXAMPLE

//**@version=**6

strategy("risk.max\_intraday\_filled\_orders Demo")

strategy.risk.max\_intraday\_filled\_orders(10) // After 10 orders are filled, no more strategy orders will be placed (except for a market order to exit current open market position, if there is any).

if open > close

strategy.entry("buy", strategy.long)

if open < close

strategy.entry("sell", strategy.short)

**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

strategy.risk.max\_intraday\_loss(value, type, alert\_message) → void

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.

**type (simple string)** A required parameter. The type of the value. Please specify one of the following values: [strategy.percent\_of\_equity](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.percent_of_equity) or [strategy.cash](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.cash). Note: if equity drops down to zero or to a negative and the [strategy.percent\_of\_equity](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.percent_of_equity) is specified, all pending orders are cancelled, all open positions are closed and no new orders can be placed for good.

**alert\_message (simple string)** An optional parameter which replaces the {{strategy.order.alert\_message}} placeholder when it is used in the "Create Alert" dialog box's "Message" field.

EXAMPLE

// Sets the maximum intraday loss using the strategy's equity value.

//**@version=**6

strategy("strategy.risk.max\_intraday\_loss Example 1", overlay = false, default\_qty\_type = strategy.percent\_of\_equity, default\_qty\_value = 100)

// Input for maximum intraday loss %.

lossPct = input.float(10)

// Set maximum intraday loss to our lossPct input

strategy.risk.max\_intraday\_loss(lossPct, strategy.percent\_of\_equity)

// Enter Short at bar\_index zero.

if bar\_index == 0

strategy.entry("Short", strategy.short)

// Store equity value from the beginning of the day

eqFromDayStart = ta.valuewhen(ta.change(dayofweek) > 0, strategy.equity, 0)

// Calculate change of the current equity from the beginning of the current day.

eqChgPct = 100 \* ((strategy.equity - eqFromDayStart) / strategy.equity)

// Plot it

plot(eqChgPct)

hline(-lossPct)

EXAMPLE

// Sets the maximum intraday loss using the strategy's cash value.

//**@version=**6

strategy("strategy.risk.max\_intraday\_loss Example 2", overlay = false)

// Input for maximum intraday loss in absolute cash value of the symbol.

absCashLoss = input.float(5)

// Set maximum intraday loss to `absCashLoss` in account currency.

strategy.risk.max\_intraday\_loss(absCashLoss, strategy.cash)

// Enter Short at bar\_index zero.

if bar\_index == 0

strategy.entry("Short", strategy.short)

// Store the open price value from the beginning of the day.

beginPrice = ta.valuewhen(ta.change(dayofweek) > 0, open, 0)

// Calculate the absolute price change for the current period.

priceChg = (close - beginPrice)

hline(absCashLoss)

plot(priceChg)

SEE ALSO

[strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy)[strategy.percent\_of\_equity](https://www.tradingview.com/pine-script-reference/v6/#const_strategy.percent_of_equity)[strategy.cash](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_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

strategy.risk.max\_position\_size(contracts) → void

ARGUMENTS

**contracts (simple int/float)** A required parameter. Maximum number of contracts/shares/lots/units in a position.

EXAMPLE

//**@version=**6

strategy("risk.max\_position\_size Demo", default\_qty\_value = 100)

strategy.risk.max\_position\_size(10)

if open > close

strategy.entry("buy", strategy.long)

plot(strategy.position\_size) // max plot value will be 10

**string()**4 overloads

Casts na to string

SYNTAX & OVERLOADS

[string(x) → const string](https://www.tradingview.com/pine-script-reference/v6/#fun_string-0)

[string(x) → input string](https://www.tradingview.com/pine-script-reference/v6/#fun_string-1)

[string(x) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_string-2)

[string(x) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_string-3)

ARGUMENTS

**x (const string)** The value to convert to the specified type, usually [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

RETURNS

The value of the argument after casting to string.

SEE ALSO

[float](https://www.tradingview.com/pine-script-reference/v6/#fun_float)[int](https://www.tradingview.com/pine-script-reference/v6/#fun_int)[bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool)[color](https://www.tradingview.com/pine-script-reference/v6/#fun_color)[line](https://www.tradingview.com/pine-script-reference/v6/#fun_line)[label](https://www.tradingview.com/pine-script-reference/v6/#fun_label)

**syminfo.prefix()**2 overloads

Returns exchange prefix of the symbol, e.g. "NASDAQ".

SYNTAX & OVERLOADS

[syminfo.prefix(symbol) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_syminfo.prefix-0)

[syminfo.prefix(symbol) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_syminfo.prefix-1)

ARGUMENTS

**symbol (simple string)** Symbol. Note that the symbol should be passed with a prefix. For example: "NASDAQ:AAPL" instead of "AAPL".

EXAMPLE

//**@version=**6

indicator("syminfo.prefix fun", overlay=true)

i\_sym = input.symbol("NASDAQ:AAPL")

pref = syminfo.prefix(i\_sym)

tick = syminfo.ticker(i\_sym)

t = ticker.new(pref, tick, session.extended)

s = request.security(t, "1D", close)

plot(s)

RETURNS

Returns exchange prefix of the symbol, e.g. "NASDAQ".

REMARKS

The result of the function is used in the [ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)/[ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify) and [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security).

SEE ALSO

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.prefix](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.prefix)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#fun_syminfo.ticker)[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)

**syminfo.ticker()**2 overloads

Returns symbol name without exchange prefix, e.g. "AAPL".

SYNTAX & OVERLOADS

[syminfo.ticker(symbol) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_syminfo.ticker-0)

[syminfo.ticker(symbol) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_syminfo.ticker-1)

ARGUMENTS

**symbol (simple string)** Symbol. Note that the symbol should be passed with a prefix. For example: "NASDAQ:AAPL" instead of "AAPL".

EXAMPLE

//**@version=**6

indicator("syminfo.ticker fun", overlay=true)

i\_sym = input.symbol("NASDAQ:AAPL")

pref = syminfo.prefix(i\_sym)

tick = syminfo.ticker(i\_sym)

t = ticker.new(pref, tick, session.extended)

s = request.security(t, "1D", close)

plot(s)

RETURNS

Returns symbol name without exchange prefix, e.g. "AAPL".

REMARKS

The result of the function is used in the [ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)/[ticker.modify](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify) and [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security).

SEE ALSO

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.prefix](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.prefix)[syminfo.prefix](https://www.tradingview.com/pine-script-reference/v6/#fun_syminfo.prefix)[ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new)

**ta.alma()**2 overloads

Arnaud Legoux Moving Average. It uses Gaussian distribution as weights for moving average.

SYNTAX & OVERLOADS

[ta.alma(series, length, offset, sigma) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.alma-0)

[ta.alma(series, length, offset, sigma, floor) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.alma-1)

ARGUMENTS

**series (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

**offset (simple int/float)** Controls tradeoff between smoothness (closer to 1) and responsiveness (closer to 0).

**sigma (simple int/float)** Changes the smoothness of ALMA. The larger sigma the smoother ALMA.

EXAMPLE

//**@version=**6

indicator("ta.alma", overlay=true)

plot(ta.alma(close, 9, 0.85, 6))

// same on pine, but much less efficient

pine\_alma(series, windowsize, offset, sigma) =>

m = offset \* (windowsize - 1)

//m = math.floor(offset \* (windowsize - 1)) // Used as m when math.floor=true

s = windowsize / sigma

norm = 0.0

sum = 0.0

for i = 0 to windowsize - 1

weight = math.exp(-1 \* math.pow(i - m, 2) / (2 \* math.pow(s, 2)))

norm := norm + weight

sum := sum + **series**[windowsize - i - 1] \* weight

sum / norm

plot(pine\_alma(close, 9, 0.85, 6))

RETURNS

Arnaud Legoux Moving Average.

REMARKS

na values in the source series are included in calculations and will produce an na result.

SEE ALSO

[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)[ta.ema](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.ema)[ta.rma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rma)[ta.wma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.wma)[ta.vwma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwma)[ta.swma](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.atr(length) → series float

ARGUMENTS

**length (simple int)** Length (number of bars back).

EXAMPLE

//**@version=**6

indicator("ta.atr")

plot(ta.atr(14))

//the same on pine

pine\_atr(length) =>

trueRange = na(high[1])? high-low : math.max(math.max(high - low, math.abs(high - close[1])), math.abs(low - close[1]))

//true range can be also calculated with ta.tr(true)

ta.rma(trueRange, length)

plot(pine\_atr(14))

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

[ta.tr](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.tr)[ta.rma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rma)

**ta.barssince()**

Counts the number of bars since the last time the condition was true.

SYNTAX

ta.barssince(condition) → series int

ARGUMENTS

**condition (series bool)** The condition to check for.

EXAMPLE

//**@version=**6

indicator("ta.barssince")

// get number of bars since last color.green bar

plot(ta.barssince(close >= open))

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.

Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[ta.lowestbars](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowestbars)[ta.highestbars](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highestbars)[ta.valuewhen](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.valuewhen)[ta.highest](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highest)[ta.lowest](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.bb(series, length, mult) → [series float, series float, series float]

ARGUMENTS

**series (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

**mult (simple int/float)** Standard deviation factor.

EXAMPLE

//**@version=**6

indicator("ta.bb")

[middle, upper, lower] = ta.bb(close, 5, 4)

plot(middle, color=color.yellow)

plot(upper, color=color.yellow)

plot(lower, color=color.yellow)

// the same on pine

f\_bb(src, length, mult) =>

**float** basis = ta.sma(src, length)

**float** dev = mult \* ta.stdev(src, length)

[basis, basis + dev, basis - dev]

[pineMiddle, pineUpper, pineLower] = f\_bb(close, 5, 4)

plot(pineMiddle)

plot(pineUpper)

plot(pineLower)

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

[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)[ta.stdev](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.stdev)[ta.kc](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.bbw(series, length, mult) → series float

ARGUMENTS

**series (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

**mult (simple int/float)** Standard deviation factor.

EXAMPLE

//**@version=**6

indicator("ta.bbw")

plot(ta.bbw(close, 5, 4), color=color.yellow)

// the same on pine

f\_bbw(src, length, mult) =>

**float** basis = ta.sma(src, length)

**float** dev = mult \* ta.stdev(src, length)

(((basis + dev) - (basis - dev)) / basis) \* 100

plot(f\_bbw(close, 5, 4))

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

[ta.bb](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.bb)[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)[ta.stdev](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.cci(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

RETURNS

Commodity channel index of source for length bars back.

REMARKS

na values in the source series are ignored.

**ta.change()**6 overloads

Compares the current source value to its value length bars ago and returns the difference.

SYNTAX & OVERLOADS

[ta.change(source) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.change-0)

[ta.change(source) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.change-1)

[ta.change(source, length) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.change-2)

[ta.change(source, length) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.change-3)

[ta.change(source) → series bool](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.change-4)

[ta.change(source, length) → series bool](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.change-5)

ARGUMENTS

**source (series int)** Source series.

EXAMPLE

//**@version=**6

indicator('Day and Direction Change', overlay = true)

dailyBarTime = time('1D')

isNewDay = ta.change(dailyBarTime) != 0

bgcolor(isNewDay ? color.new(color.green, 80) : na)

isGreenBar = close >= open

colorChange = ta.change(isGreenBar)

plotshape(colorChange, 'Direction Change')

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

[ta.mom](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.mom)[ta.cross](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.cmo(series, length) → series float

ARGUMENTS

**series (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

EXAMPLE

//**@version=**6

indicator("ta.cmo")

plot(ta.cmo(close, 5), color=color.yellow)

// the same on pine

f\_cmo(src, length) =>

**float** mom = ta.change(src)

**float** sm1 = math.sum((mom >= 0) ? mom : 0.0, length)

**float** sm2 = math.sum((mom >= 0) ? 0.0 : -mom, length)

100 \* (sm1 - sm2) / (sm1 + sm2)

plot(f\_cmo(close, 5))

RETURNS

Chande Momentum Oscillator.

REMARKS

na values in the source series are ignored.

SEE ALSO

[ta.rsi](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rsi)[ta.stoch](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.stoch)[math.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sum)

**ta.cog()**

The cog (center of gravity) is an indicator based on statistics and the Fibonacci golden ratio.

SYNTAX

ta.cog(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

EXAMPLE

//**@version=**6

indicator("ta.cog", overlay=true)

plot(ta.cog(close, 10))

// the same on pine

pine\_cog(source, length) =>

sum = math.sum(source, length)

num = 0.0

for i = 0 to length - 1

price = source[i]

num := num + price \* (i + 1)

-num / sum

plot(pine\_cog(close, 10))

RETURNS

Center of Gravity.

REMARKS

na values in the source series are ignored.

SEE ALSO

[ta.stoch](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.stoch)

**ta.correlation()**

Correlation coefficient. Describes the degree to which two series tend to deviate from their [ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma) values.

SYNTAX

ta.correlation(source1, source2, length) → series float

ARGUMENTS

**source1 (series int/float)** Source series.

**source2 (series int/float)** Target series.

**length (series int)** Length (number of bars back).

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

[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)

**ta.cross()**

SYNTAX

ta.cross(source1, source2) → series bool

ARGUMENTS

**source1 (series int/float)** First data series.

**source2 (series int/float)** Second data series.

RETURNS

true if two series have crossed each other, otherwise false.

SEE ALSO

[ta.change](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.crossover(source1, source2) → series bool

ARGUMENTS

**source1 (series int/float)** First data series.

**source2 (series int/float)** Second 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

ta.crossunder(source1, source2) → series bool

ARGUMENTS

**source1 (series int/float)** First data series.

**source2 (series int/float)** Second 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

ta.cum(source) → series float

ARGUMENTS

**source (series int/float)** Source used for the calculation.

RETURNS

Total sum series.

SEE ALSO

[math.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sum)

**ta.dev()**

Measure of difference between the series and it's [ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)

SYNTAX

ta.dev(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

EXAMPLE

//**@version=**6

indicator("ta.dev")

plot(ta.dev(close, 10))

// the same on pine

pine\_dev(source, length) =>

mean = ta.sma(source, length)

sum = 0.0

for i = 0 to length - 1

val = source[i]

sum := sum + math.abs(val - mean)

dev = sum/length

plot(pine\_dev(close, 10))

RETURNS

Deviation of source for length bars back.

REMARKS

na values in the source series are ignored.

SEE ALSO

[ta.variance](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.variance)[ta.stdev](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.stdev)

**ta.dmi()**

The dmi function returns the directional movement index.

SYNTAX

ta.dmi(diLength, adxSmoothing) → [series float, series float, series float]

ARGUMENTS

**diLength (simple int)** DI Period.

**adxSmoothing (simple int)** ADX Smoothing Period.

EXAMPLE

//**@version=**6

indicator(title="Directional Movement Index", shorttitle="DMI", format=format.price, precision=4)

len = input.int(17, minval=1, title="DI Length")

lensig = input.int(14, title="ADX Smoothing", minval=1)

[diplus, diminus, adx] = ta.dmi(len, lensig)

plot(adx, color=color.red, title="ADX")

plot(diplus, color=color.blue, title="+DI")

plot(diminus, color=color.orange, title="-DI")

RETURNS

[Tuple](https://www.tradingview.com/pine-script-docs/language/type-system/#tuples) of three DMI series: Positive Directional Movement (+DI), Negative Directional Movement (-DI) and Average Directional Movement Index (ADX).

SEE ALSO

[ta.rsi](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rsi)[ta.tsi](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.tsi)[ta.mfi](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.ema(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (simple int)** Number of bars (length).

EXAMPLE

//**@version=**6

indicator("ta.ema")

plot(ta.ema(close, 15))

//the same on pine

pine\_ema(src, length) =>

alpha = 2 / (length + 1)

sum = 0.0

sum := na(sum[1]) ? src : alpha \* src + (1 - alpha) \* nz(sum[1])

plot(pine\_ema(close,15))

RETURNS

Exponential moving average of source with alpha = 2 / (length + 1).

REMARKS

Please note that using this variable/function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

na values in the source series are ignored; the function calculates on the length quantity of non-na values.

SEE ALSO

[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)[ta.rma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rma)[ta.wma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.wma)[ta.vwma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwma)[ta.swma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.swma)[ta.alma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.alma)

**ta.falling()**

Test if the source series is now falling for length bars long.

SYNTAX

ta.falling(source, length) → series bool

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

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

[ta.rising](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rising)

**ta.highest()**2 overloads

Highest value for a given number of bars back.

SYNTAX & OVERLOADS

[ta.highest(length) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highest-0)

[ta.highest(source, length) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highest-1)

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.

One arg version: length is the number of bars back. Algorithm uses high as a source series.

na values in the source series are ignored.

SEE ALSO

[ta.lowest](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowest)[ta.lowestbars](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowestbars)[ta.highestbars](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highestbars)[ta.valuewhen](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.valuewhen)[ta.barssince](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.barssince)

**ta.highestbars()**2 overloads

Highest value offset for a given number of bars back.

SYNTAX & OVERLOADS

[ta.highestbars(length) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highestbars-0)

[ta.highestbars(source, length) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highestbars-1)

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.

One arg version: length is the number of bars back. Algorithm uses high as a source series.

na values in the source series are ignored.

SEE ALSO

[ta.lowest](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowest)[ta.highest](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highest)[ta.lowestbars](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowestbars)[ta.barssince](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.barssince)[ta.valuewhen](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.valuewhen)

**ta.hma()**

The hma function returns the Hull Moving Average.

SYNTAX

ta.hma(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (simple int)** Number of bars.

EXAMPLE

//**@version=**6

indicator("Hull Moving Average")

src = input(defval=close, title="Source")

length = input(defval=9, title="Length")

hmaBuildIn = ta.hma(src, length)

plot(hmaBuildIn, title="Hull MA", color=#674EA7)

RETURNS

Hull moving average of 'source' for 'length' bars back.

REMARKS

na values in the source series are ignored.

SEE ALSO

[ta.ema](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.ema)[ta.rma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rma)[ta.wma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.wma)[ta.vwma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwma)[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)

**ta.kc()**2 overloads

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

[ta.kc(series, length, mult) → [series float, series float, series float]](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.kc-0)

[ta.kc(series, length, mult, useTrueRange) → [series float, series float, series float]](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.kc-1)

ARGUMENTS

**series (series int/float)** Series of values to process.

**length (simple int)** Number of bars (length).

**mult (simple int/float)** Standard deviation factor.

EXAMPLE

//**@version=**6

indicator("ta.kc")

[middle, upper, lower] = ta.kc(close, 5, 4)

plot(middle, color=color.yellow)

plot(upper, color=color.yellow)

plot(lower, color=color.yellow)

// the same on pine

f\_kc(src, length, mult, useTrueRange) =>

**float** basis = ta.ema(src, length)

**float** span = (useTrueRange) ? ta.tr : (high - low)

**float** rangeEma = ta.ema(span, length)

[basis, basis + rangeEma \* mult, basis - rangeEma \* mult]

[pineMiddle, pineUpper, pineLower] = f\_kc(close, 5, 4, true)

plot(pineMiddle)

plot(pineUpper)

plot(pineLower)

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

[ta.ema](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.ema)[ta.atr](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.atr)[ta.bb](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.bb)

**ta.kcw()**2 overloads

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

[ta.kcw(series, length, mult) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.kcw-0)

[ta.kcw(series, length, mult, useTrueRange) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.kcw-1)

ARGUMENTS

**series (series int/float)** Series of values to process.

**length (simple int)** Number of bars (length).

**mult (simple int/float)** Standard deviation factor.

EXAMPLE

//**@version=**6

indicator("ta.kcw")

plot(ta.kcw(close, 5, 4), color=color.yellow)

// the same on pine

f\_kcw(src, length, mult, useTrueRange) =>

**float** basis = ta.ema(src, length)

**float** span = (useTrueRange) ? ta.tr : (high - low)

**float** rangeEma = ta.ema(span, length)

((basis + rangeEma \* mult) - (basis - rangeEma \* mult)) / basis

plot(f\_kcw(close, 5, 4, true))

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

[ta.kc](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.kc)[ta.ema](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.ema)[ta.atr](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.atr)[ta.bb](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.linreg(source, length, offset) → series float

ARGUMENTS

**source (series int/float)** Source series.

**length (series int)** Number of bars (length).

**offset (simple int)** Offset.

RETURNS

Linear regression curve.

REMARKS

na values in the source series are included in calculations and will produce an na result.

**ta.lowest()**2 overloads

Lowest value for a given number of bars back.

SYNTAX & OVERLOADS

[ta.lowest(length) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowest-0)

[ta.lowest(source, length) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowest-1)

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.

One arg version: length is the number of bars back. Algorithm uses low as a source series.

na values in the source series are ignored.

SEE ALSO

[ta.highest](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highest)[ta.lowestbars](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowestbars)[ta.highestbars](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highestbars)[ta.valuewhen](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.valuewhen)[ta.barssince](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.barssince)

**ta.lowestbars()**2 overloads

Lowest value offset for a given number of bars back.

SYNTAX & OVERLOADS

[ta.lowestbars(length) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowestbars-0)

[ta.lowestbars(source, length) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowestbars-1)

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.

One arg version: length is the number of bars back. Algorithm uses low as a source series.

na values in the source series are ignored.

SEE ALSO

[ta.lowest](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowest)[ta.highest](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highest)[ta.highestbars](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highestbars)[ta.barssince](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.barssince)[ta.valuewhen](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.macd(source, fastlen, slowlen, siglen) → [series float, series float, series float]

ARGUMENTS

**source (series int/float)** Series of values to process.

**fastlen (simple int)** Fast Length parameter.

**slowlen (simple int)** Slow Length parameter.

**siglen (simple int)** Signal Length parameter.

EXAMPLE

//**@version=**6

indicator("MACD")

[macdLine, signalLine, histLine] = ta.macd(close, 12, 26, 9)

plot(macdLine, color=color.blue)

plot(signalLine, color=color.orange)

plot(histLine, color=color.red, style=plot.style\_histogram)

If you need only one value, use placeholders '\_' like this:

EXAMPLE

//**@version=**6

indicator("MACD")

[\_, signalLine, \_] = ta.macd(close, 12, 26, 9)

plot(signalLine, color=color.orange)

RETURNS

[Tuple](https://www.tradingview.com/pine-script-docs/language/type-system/#tuples) 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

[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)[ta.ema](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.ema)

**ta.max()**

Returns the all-time high value of source from the beginning of the chart up to the current bar.

SYNTAX

ta.max(source) → series float

ARGUMENTS

**source (series int/float)** Source used for the calculation.

REMARKS

[na](https://www.tradingview.com/pine-script-reference/v6/#var_na) occurrences of source are ignored.

**ta.median()**2 overloads

Returns the median of the series.

SYNTAX & OVERLOADS

[ta.median(source, length) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.median-0)

[ta.median(source, length) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.median-1)

ARGUMENTS

**source (series int)** Series of values to process.

**length (series int)** Number of bars (length).

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

ta.mfi(series, length) → series float

ARGUMENTS

**series (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

EXAMPLE

//**@version=**6

indicator("Money Flow Index")

plot(ta.mfi(hlc3, 14), color=color.yellow)

// the same on pine

pine\_mfi(src, length) =>

**float** upper = math.sum(volume \* (ta.change(src) <= 0.0 ? 0.0 : src), length)

**float** lower = math.sum(volume \* (ta.change(src) >= 0.0 ? 0.0 : src), length)

mfi = 100.0 - (100.0 / (1.0 + upper / lower))

mfi

plot(pine\_mfi(hlc3, 14))

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

[ta.rsi](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rsi)[math.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_math.sum)

**ta.min()**

Returns the all-time low value of source from the beginning of the chart up to the current bar.

SYNTAX

ta.min(source) → series float

ARGUMENTS

**source (series int/float)** Source used for the calculation.

REMARKS

[na](https://www.tradingview.com/pine-script-reference/v6/#var_na) occurrences of source are ignored.

**ta.mode()**2 overloads

Returns the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the series. If there are several values with the same frequency, it returns the smallest value.

SYNTAX & OVERLOADS

[ta.mode(source, length) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.mode-0)

[ta.mode(source, length) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.mode-1)

ARGUMENTS

**source (series int)** Series of values to process.

**length (series int)** Number of bars (length).

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

ta.mom(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Offset from the current bar to the previous bar.

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

[ta.change](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.change)

**ta.percentile\_linear\_interpolation()**

Calculates percentile using method of linear interpolation between the two nearest ranks.

SYNTAX

ta.percentile\_linear\_interpolation(source, length, percentage) → series float

ARGUMENTS

**source (series int/float)** Series of values to process (source).

**length (series int)** Number of bars back (length).

**percentage (simple int/float)** Percentage, a number from range 0..100.

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.

na values in the source series are included in calculations and will produce an na result.

SEE ALSO

[ta.percentile\_nearest\_rank](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.percentile_nearest_rank)

**ta.percentile\_nearest\_rank()**

Calculates percentile using method of Nearest Rank.

SYNTAX

ta.percentile\_nearest\_rank(source, length, percentage) → series float

ARGUMENTS

**source (series int/float)** Series of values to process (source).

**length (series int)** Number of bars back (length).

**percentage (simple int/float)** Percentage, a number from range 0..100.

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.

A percentile calculated using the Nearest Rank method will always be a member of the input data set.

The 100th percentile is defined to be the largest value in the input data set.

na values in the source series are ignored.

SEE ALSO

[ta.percentile\_linear\_interpolation](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.percentrank(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

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

ta.pivot\_point\_levels(type, anchor, developing) → array<float>

ARGUMENTS

**type (series string)** The type of pivot point levels. Possible values: "Traditional", "Fibonacci", "Woodie", "Classic", "DM", "Camarilla".

**anchor (series bool)** The condition that triggers the reset of the pivot point calculations. When [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), calculations reset; when [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), results calculated at the last reset persist.

**developing (series bool)** If [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), the values are those calculated the last time the anchor condition was [true](https://www.tradingview.com/pine-script-reference/v6/#const_true). They remain constant until the anchor condition becomes [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) again. If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the pivots are developing, i.e., they constantly recalculate on the data developing between the point of the last anchor (or bar zero if the anchor condition was never [true](https://www.tradingview.com/pine-script-reference/v6/#const_true)) and the current bar. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("Weekly Pivots", max\_lines\_count=500, overlay=true)

timeframe = "1W"

typeInput = input.string("Traditional", "Type", options=["Traditional", "Fibonacci", "Woodie", "Classic", "DM", "Camarilla"])

weekChange = timeframe.change(timeframe)

pivotPointsArray = ta.pivot\_point\_levels(typeInput, weekChange)

if weekChange

for pivotLevel in pivotPointsArray

line.new(time, pivotLevel, time + timeframe.in\_seconds(timeframe) \* 1000, pivotLevel, xloc=xloc.bar\_time)

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](https://www.tradingview.com/pine-script-reference/v6/#var_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()**2 overloads

This function returns price of the pivot high point. It returns 'NaN', if there was no pivot high point.

SYNTAX & OVERLOADS

[ta.pivothigh(leftbars, rightbars) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.pivothigh-0)

[ta.pivothigh(source, leftbars, rightbars) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.pivothigh-1)

ARGUMENTS

**leftbars (series int/float)** Left strength.

**rightbars (series int/float)** Right strength.

EXAMPLE

//**@version=**6

indicator("PivotHigh", overlay=true)

leftBars = input(2)

rightBars=input(2)

ph = ta.pivothigh(leftBars, rightBars)

plot(ph, style=plot.style\_cross, linewidth=3, color= color.red, offset=-rightBars)

RETURNS

Price of the point or 'NaN'.

REMARKS

If parameters 'leftbars' or 'rightbars' are series you should use [max\_bars\_back](https://www.tradingview.com/pine-script-reference/v6/#fun_max_bars_back) function for the 'source' variable.

**ta.pivotlow()**2 overloads

This function returns price of the pivot low point. It returns 'NaN', if there was no pivot low point.

SYNTAX & OVERLOADS

[ta.pivotlow(leftbars, rightbars) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.pivotlow-0)

[ta.pivotlow(source, leftbars, rightbars) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.pivotlow-1)

ARGUMENTS

**leftbars (series int/float)** Left strength.

**rightbars (series int/float)** Right strength.

EXAMPLE

//**@version=**6

indicator("PivotLow", overlay=true)

leftBars = input(2)

rightBars=input(2)

pl = ta.pivotlow(close, leftBars, rightBars)

plot(pl, style=plot.style\_cross, linewidth=3, color= color.blue, offset=-rightBars)

RETURNS

Price of the point or 'NaN'.

REMARKS

If parameters 'leftbars' or 'rightbars' are series you should use [max\_bars\_back](https://www.tradingview.com/pine-script-reference/v6/#fun_max_bars_back) function for the 'source' variable.

**ta.range()**2 overloads

Returns the difference between the min and max values in a series.

SYNTAX & OVERLOADS

[ta.range(source, length) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.range-0)

[ta.range(source, length) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.range-1)

ARGUMENTS

**source (series int)** Series of values to process.

**length (series int)** Number of bars (length).

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

ta.rci(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (simple int)** Number of bars (length).

RETURNS

The Rank Correlation Index, a value between -100 to 100.

SEE ALSO

**ta.rising()**

Test if the source series is now rising for length bars long.

SYNTAX

ta.rising(source, length) → series bool

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

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

[ta.falling](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.falling)

**ta.rma()**

Moving average used in RSI. It is the exponentially weighted moving average with alpha = 1 / length.

SYNTAX

ta.rma(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (simple int)** Number of bars (length).

EXAMPLE

//**@version=**6

indicator("ta.rma")

plot(ta.rma(close, 15))

//the same on pine

pine\_rma(src, length) =>

alpha = 1/length

sum = 0.0

sum := na(sum[1]) ? ta.sma(src, length) : alpha \* src + (1 - alpha) \* nz(sum[1])

plot(pine\_rma(close, 15))

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

[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)[ta.ema](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.ema)[ta.wma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.wma)[ta.vwma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwma)[ta.swma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.swma)[ta.alma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.alma)[ta.rsi](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rsi)

**ta.roc()**

Calculates the percentage of change (rate of change) between the current value of source and its value length bars ago.

It is calculated by the formula: 100 \* change(src, length) / src[length].

SYNTAX

ta.roc(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

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

ta.rsi(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (simple int)** Number of bars (length).

EXAMPLE

//**@version=**6

indicator("ta.rsi")

plot(ta.rsi(close, 7))

// same on pine, but less efficient

pine\_rsi(x, y) =>

u = math.max(x - x[1], 0) // upward ta.change

d = math.max(x[1] - x, 0) // downward ta.change

rs = ta.rma(u, y) / ta.rma(d, y)

res = 100 - 100 / (1 + rs)

res

plot(pine\_rsi(close, 7))

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

[ta.rma](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.sar(start, inc, max) → series float

ARGUMENTS

**start (simple int/float)** Start.

**inc (simple int/float)** Increment.

**max (simple int/float)** Maximum.

EXAMPLE

//**@version=**6

indicator("ta.sar")

plot(ta.sar(0.02, 0.02, 0.2), style=plot.style\_cross, linewidth=3)

// The same on Pine Script®

pine\_sar(start, inc, max) =>

var **float** result = na

var **float** maxMin = na

var **float** acceleration = na

var **bool** isBelow = false

**bool** isFirstTrendBar = false

if bar\_index == 1

if close > close[1]

isBelow := true

maxMin := high

result := low[1]

else

isBelow := false

maxMin := low

result := high[1]

isFirstTrendBar := true

acceleration := start

result := result + acceleration \* (maxMin - result)

if isBelow

if result > low

isFirstTrendBar := true

isBelow := false

result := math.max(high, maxMin)

maxMin := low

acceleration := start

else

if result < high

isFirstTrendBar := true

isBelow := true

result := math.min(low, maxMin)

maxMin := high

acceleration := start

if not isFirstTrendBar

if isBelow

if high > maxMin

maxMin := high

acceleration := math.min(acceleration + inc, max)

else

if low < maxMin

maxMin := low

acceleration := math.min(acceleration + inc, max)

if isBelow

result := math.min(result, low[1])

if bar\_index > 1

result := math.min(result, low[2])

else

result := math.max(result, high[1])

if bar\_index > 1

result := math.max(result, high[2])

result

plot(pine\_sar(0.02, 0.02, 0.2), style=plot.style\_cross, linewidth=3)

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

ta.sma(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

EXAMPLE

//**@version=**6

indicator("ta.sma")

plot(ta.sma(close, 15))

// same on pine, but much less efficient

pine\_sma(x, y) =>

sum = 0.0

for i = 0 to y - 1

sum := sum + x[i] / y

sum

plot(pine\_sma(close, 15))

RETURNS

Simple moving average of source for length bars back.

REMARKS

na values in the source series are ignored.

SEE ALSO

[ta.ema](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.ema)[ta.rma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rma)[ta.wma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.wma)[ta.vwma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwma)[ta.swma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.swma)[ta.alma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.alma)

**ta.stdev()**

SYNTAX

ta.stdev(source, length, biased) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

**biased (series bool)** Determines which estimate should be used. Optional. The default is true.

EXAMPLE

//**@version=**6

indicator("ta.stdev")

plot(ta.stdev(close, 5))

//the same on pine

isZero(val, eps) => math.abs(val) <= eps

SUM(fst, snd) =>

EPS = 1e-10

res = fst + snd

if isZero(res, EPS)

res := 0

else

if not isZero(res, 1e-4)

res := res

else

15

pine\_stdev(src, length) =>

avg = ta.sma(src, length)

sumOfSquareDeviations = 0.0

for i = 0 to length - 1

sum = SUM(src[i], -avg)

sumOfSquareDeviations := sumOfSquareDeviations + sum \* sum

stdev = math.sqrt(sumOfSquareDeviations / length)

plot(pine\_stdev(close, 5))

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.

na values in the source series are ignored; the function calculates on the length quantity of non-na values.

SEE ALSO

[ta.dev](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.dev)[ta.variance](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.variance)

**ta.stoch()**

Stochastic. It is calculated by a formula: 100 \* (close - lowest(low, length)) / (highest(high, length) - lowest(low, length)).

SYNTAX

ta.stoch(source, high, low, length) → series float

ARGUMENTS

**source (series int/float)** Source series.

**high (series int/float)** Series of high.

**low (series int/float)** Series of low.

**length (series int)** Length (number of bars back).

RETURNS

Stochastic.

REMARKS

na values in the source series are ignored.

SEE ALSO

[ta.cog](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.cog)

**ta.supertrend()**

The Supertrend Indicator. The Supertrend is a trend following indicator.

SYNTAX

ta.supertrend(factor, atrPeriod) → [series float, series float]

ARGUMENTS

**factor (series int/float)** The multiplier by which the ATR will get multiplied.

**atrPeriod (simple int)** Length of ATR.

EXAMPLE

//**@version=**6

indicator("Pine Script® Supertrend")

[supertrend, direction] = ta.supertrend(3, 10)

plot(direction < 0 ? supertrend : na, "Up direction", color = color.green, style=plot.style\_linebr)

plot(direction > 0 ? supertrend : na, "Down direction", color = color.red, style=plot.style\_linebr)

// The same on Pine Script®

pine\_supertrend(factor, atrPeriod) =>

src = hl2

atr = ta.atr(atrPeriod)

upperBand = src + factor \* atr

lowerBand = src - factor \* atr

prevLowerBand = nz(lowerBand[1])

prevUpperBand = nz(upperBand[1])

lowerBand := lowerBand > prevLowerBand or close[1] < prevLowerBand ? lowerBand : prevLowerBand

upperBand := upperBand < prevUpperBand or close[1] > prevUpperBand ? upperBand : prevUpperBand

**int** \_direction = na

**float** superTrend = na

prevSuperTrend = superTrend[1]

if na(atr[1])

\_direction := 1

else if prevSuperTrend == prevUpperBand

\_direction := close > upperBand ? -1 : 1

else

\_direction := close < lowerBand ? 1 : -1

superTrend := \_direction == -1 ? lowerBand : upperBand

[superTrend, \_direction]

[Pine\_Supertrend, pineDirection] = pine\_supertrend(3, 10)

plot(pineDirection < 0 ? Pine\_Supertrend : na, "Up direction", color = color.green, style=plot.style\_linebr)

plot(pineDirection > 0 ? Pine\_Supertrend : na, "Down direction", color = color.red, style=plot.style\_linebr)

RETURNS

[Tuple](https://www.tradingview.com/pine-script-docs/language/type-system/#tuples) of two supertrend series: supertrend line and direction of trend. Possible values are 1 (down direction) and -1 (up direction).

SEE ALSO

[ta.macd](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.macd)

**ta.swma()**

Symmetrically weighted moving average with fixed length: 4. Weights: [1/6, 2/6, 2/6, 1/6].

SYNTAX

ta.swma(source) → series float

ARGUMENTS

**source (series int/float)** Source series.

EXAMPLE

//**@version=**6

indicator("ta.swma")

plot(ta.swma(close))

// same on pine, but less efficient

pine\_swma(x) =>

x[3] \* 1 / 6 + x[2] \* 2 / 6 + x[1] \* 2 / 6 + x[0] \* 1 / 6

plot(pine\_swma(close))

RETURNS

Symmetrically weighted moving average.

REMARKS

na values in the source series are included in calculations and will produce an na result.

SEE ALSO

[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)[ta.ema](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.ema)[ta.rma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rma)[ta.wma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.wma)[ta.vwma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwma)[ta.alma](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#var_close) to the current bar's [high](https://www.tradingview.com/pine-script-reference/v6/#var_high) and [low](https://www.tradingview.com/pine-script-reference/v6/#var_low). The formula is: math.max(high - low, math.abs(high - close[1]), math.abs(low - close[1]))

SYNTAX

ta.tr(handle\_na) → series float

ARGUMENTS

**handle\_na (simple bool)** Defines how the function calculates the result when the previous bar's [close](https://www.tradingview.com/pine-script-reference/v6/#var_close) is [na](https://www.tradingview.com/pine-script-reference/v6/#var_na). If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the function returns the bar's high - low value. If [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), it returns [na](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_ta.tr).

SEE ALSO

[ta.tr](https://www.tradingview.com/pine-script-reference/v6/#var_ta.tr)[ta.atr](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.atr)

**ta.tsi()**

True strength index. It uses moving averages of the underlying momentum of a financial instrument.

SYNTAX

ta.tsi(source, short\_length, long\_length) → series float

ARGUMENTS

**source (series int/float)** Source series.

**short\_length (simple int)** Short length.

**long\_length (simple int)** Long length.

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()**4 overloads

Returns the value of the source series on the bar where the condition was true on the nth most recent occurrence.

SYNTAX & OVERLOADS

[ta.valuewhen(condition, source, occurrence) → series color](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.valuewhen-0)

[ta.valuewhen(condition, source, occurrence) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.valuewhen-1)

[ta.valuewhen(condition, source, occurrence) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.valuewhen-2)

[ta.valuewhen(condition, source, occurrence) → series bool](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.valuewhen-3)

ARGUMENTS

**condition (series bool)** The condition to search for.

**source (series color)** The value to be returned from the bar where the condition is met.

**occurrence (simple int)** The occurrence of the condition. The numbering starts from 0 and goes back in time, so '0' is the most recent occurrence of condition, '1' is the second most recent and so forth. Must be an integer >= 0.

EXAMPLE

//**@version=**6

indicator("ta.valuewhen")

slow = ta.sma(close, 7)

fast = ta.sma(close, 14)

// Get value of `close` on second most recent cross

plot(ta.valuewhen(ta.cross(slow, fast), close, 1))

REMARKS

This function requires execution on every bar. It is not recommended to use it inside a [for](https://www.tradingview.com/pine-script-reference/v6/#kw_for) or [while](https://www.tradingview.com/pine-script-reference/v6/#kw_while) loop structure, where its behavior can be unexpected. Please note that using this function can cause [indicator repainting](https://www.tradingview.com/pine-script-docs/concepts/repainting/).

SEE ALSO

[ta.lowestbars](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowestbars)[ta.highestbars](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highestbars)[ta.barssince](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.barssince)[ta.highest](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.highest)[ta.lowest](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.lowest)

**ta.variance()**

Variance is the expectation of the squared deviation of a series from its mean ([ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)), and it informally measures how far a set of numbers are spread out from their mean.

SYNTAX

ta.variance(source, length, biased) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

**biased (series bool)** Determines which estimate should be used. Optional. The default is true.

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.

na values in the source series are ignored; the function calculates on the length quantity of non-na values.

SEE ALSO

[ta.dev](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.dev)[ta.stdev](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.stdev)

**ta.vwap()**3 overloads

Volume weighted average price.

SYNTAX & OVERLOADS

[ta.vwap(source) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwap-0)

[ta.vwap(source, anchor) → series float](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwap-1)

[ta.vwap(source, anchor, stdev\_mult) → [series float, series float, series float]](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwap-2)

ARGUMENTS

**source (series int/float)** Source used for the VWAP calculation.

EXAMPLE

//**@version=**6

indicator("Simple VWAP")

vwap = ta.vwap(open)

plot(vwap)

EXAMPLE

//**@version=**6

indicator("Advanced VWAP")

vwapAnchorInput = input.string("Daily", "Anchor", options = ["Daily", "Weekly", "Monthly"])

stdevMultiplierInput = input.float(1.0, "Standard Deviation Multiplier")

anchorTimeframe = switch vwapAnchorInput

"Daily" => "1D"

"Weekly" => "1W"

"Monthly" => "1M"

anchor = timeframe.change(anchorTimeframe)

[vwap, upper, lower] = ta.vwap(open, anchor, stdevMultiplierInput)

plot(vwap)

plot(upper, color = color.green)

plot(lower, color = color.green)

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](https://www.tradingview.com/pine-script-reference/v6/#const_true). Until then, the function returns [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

SEE ALSO

[ta.vwap](https://www.tradingview.com/pine-script-reference/v6/#var_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

ta.vwma(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

EXAMPLE

//**@version=**6

indicator("ta.vwma")

plot(ta.vwma(close, 15))

// same on pine, but less efficient

pine\_vwma(x, y) =>

ta.sma(x \* volume, y) / ta.sma(volume, y)

plot(pine\_vwma(close, 15))

RETURNS

Volume-weighted moving average of source for length bars back.

REMARKS

na values in the source series are ignored.

SEE ALSO

[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)[ta.ema](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.ema)[ta.rma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rma)[ta.wma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.wma)[ta.swma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.swma)[ta.alma](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.wma(source, length) → series float

ARGUMENTS

**source (series int/float)** Series of values to process.

**length (series int)** Number of bars (length).

EXAMPLE

//**@version=**6

indicator("ta.wma")

plot(ta.wma(close, 15))

// same on pine, but much less efficient

pine\_wma(x, y) =>

norm = 0.0

sum = 0.0

for i = 0 to y - 1

weight = (y - i) \* y

norm := norm + weight

sum := sum + x[i] \* weight

sum / norm

plot(pine\_wma(close, 15))

RETURNS

Weighted moving average of source for length bars back.

REMARKS

na values in the source series are ignored.

SEE ALSO

[ta.sma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.sma)[ta.ema](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.ema)[ta.rma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.rma)[ta.vwma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.vwma)[ta.swma](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.swma)[ta.alma](https://www.tradingview.com/pine-script-reference/v6/#fun_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

ta.wpr(length) → series float

ARGUMENTS

**length (series int)** Number of bars.

EXAMPLE

//**@version=**6

indicator("Williams %R", shorttitle="%R", format=format.price, precision=2)

plot(ta.wpr(14), title="%R", color=color.new(#ff6d00, 0))

RETURNS

Williams %R.

REMARKS

na values in the source series are ignored.

SEE ALSO

[ta.mfi](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.mfi)[ta.cmo](https://www.tradingview.com/pine-script-reference/v6/#fun_ta.cmo)

**table()**

Casts na to table

SYNTAX

table(x) → series table

ARGUMENTS

**x (series table)** The value to convert to the specified type, usually [na](https://www.tradingview.com/pine-script-reference/v6/#var_na).

RETURNS

The value of the argument after casting to table.

SEE ALSO

[float](https://www.tradingview.com/pine-script-reference/v6/#fun_float)[int](https://www.tradingview.com/pine-script-reference/v6/#fun_int)[bool](https://www.tradingview.com/pine-script-reference/v6/#fun_bool)[color](https://www.tradingview.com/pine-script-reference/v6/#fun_color)[string](https://www.tradingview.com/pine-script-reference/v6/#fun_string)[line](https://www.tradingview.com/pine-script-reference/v6/#fun_line)[label](https://www.tradingview.com/pine-script-reference/v6/#fun_label)

**table.cell()**

The function defines a cell in the table and sets its attributes.

SYNTAX

table.cell(table\_id, column, row, text, width, height, text\_color, text\_halign, text\_valign, text\_size, bgcolor, tooltip, text\_font\_family, text\_formatting) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**text (series string)** The text to be displayed inside the cell. Optional. The default is empty string.

**width (series int/float)** The width of the cell as a % of the indicator's visual space. Optional. By default, auto-adjusts the width based on the text inside the cell. Value 0 has the same effect.

**height (series int/float)** The height of the cell as a % of the indicator's visual space. Optional. By default, auto-adjusts the height based on the text inside of the cell. Value 0 has the same effect.

**text\_color (series color)** The color of the text. Optional. The default is [color.black](https://www.tradingview.com/pine-script-reference/v6/#const_color.black).

**text\_halign (series string)** The horizontal alignment of the cell's text. Optional. The default value is [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center). Possible values: [text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left), [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center), [text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_right).

**text\_valign (series string)** The vertical alignment of the cell's text. Optional. The default value is [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center). Possible values: [text.align\_top](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_top), [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center), [text.align\_bottom](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_bottom).

**text\_size (series int/string)** Size of the object. The size can be any positive integer, or one of the size.\* built-in constant strings. The constant strings and their equivalent integer values are: [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto) (0), [size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny) (8), [size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small) (10), [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal) (14), [size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large) (20), [size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge) (36). The default value is [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal) or 14.

**bgcolor (series color)** The background color of the text. Optional. The default is no color.

**tooltip (series string)** The tooltip to be displayed inside the cell. Optional.

**text\_font\_family (series string)** The font family of the text. Optional. The default value is [font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default). Possible values: [font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default), [font.family\_monospace](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_monospace).

**text\_formatting (const text\_format)** The formatting of the displayed text. Formatting options support addition. For example, text.format\_bold + text.format\_italic will make the text both bold and italicized. Possible values: [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none), [text.format\_bold](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_bold), [text.format\_italic](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_italic). Optional. The default is [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none).

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](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new).

Each [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) call overwrites all previously defined properties of a cell. If you call [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) twice in a row, e.g., the first time with text='Test Text', and the second time with text\_color=[color.red](https://www.tradingview.com/pine-script-reference/v6/#const_color.red) but without a new text argument, the default value of the 'text' being an empty string, it will overwrite 'Test Text', and your cell will display an empty string. If you want, instead, to modify any of the cell's properties, use the table.cell\_set\_\*() functions.

A single script can only display one table in each of the possible locations. If [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) is used on several bars to change the same attribute of a cell (e.g. change the background color of the cell to red on the first bar, then to yellow on the second bar), only the last change will be reflected in the table, i.e., the cell’s background will be yellow. Avoid unnecessary setting of cell properties by enclosing function calls in an [if](https://www.tradingview.com/pine-script-reference/v6/#kw_if) [barstate.islast](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islast) block whenever possible, to restrict their execution to the last bar of the series.

SEE ALSO

[table.cell\_set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_bgcolor)[table.cell\_set\_height](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_height)[table.cell\_set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text)[table.cell\_set\_text\_formatting](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_formatting)[table.cell\_set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_color)[table.cell\_set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_halign)[table.cell\_set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_size)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[table.cell\_set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_width)[table.cell\_set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_tooltip)

**table.cell\_set\_bgcolor()**

The function sets the background color of the cell.

SYNTAX

table.cell\_set\_bgcolor(table\_id, column, row, bgcolor) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**bgcolor (series color)** The background color of the cell.

SEE ALSO

[table.cell\_set\_height](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_height)[table.cell\_set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text)[table.cell\_set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_color)[table.cell\_set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_halign)[table.cell\_set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_size)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[table.cell\_set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_width)[table.cell\_set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_tooltip)

**table.cell\_set\_height()**

The function sets the height of cell.

SYNTAX

table.cell\_set\_height(table\_id, column, row, height) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**height (series int/float)** The height of the cell as a % of the chart window. Passing 0 auto-adjusts the height based on the text inside of the cell.

SEE ALSO

[table.cell\_set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_bgcolor)[table.cell\_set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text)[table.cell\_set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_color)[table.cell\_set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_halign)[table.cell\_set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_size)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[table.cell\_set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_width)[table.cell\_set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_tooltip)

**table.cell\_set\_text()**

The function sets the text in the specified cell.

SYNTAX

table.cell\_set\_text(table\_id, column, row, text) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**text (series string)** The text to be displayed inside the cell.

EXAMPLE

//**@version=**6

indicator("TABLE example")

var tLog = table.new(position = position.top\_left, rows = 1, columns = 2, bgcolor = color.yellow, border\_width=1)

table.cell(tLog, row = 0, column = 0, text = "sometext", text\_color = color.blue)

table.cell\_set\_text(tLog, row = 0, column = 0, text = "sometext")

SEE ALSO

[table.cell\_set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_bgcolor)[table.cell\_set\_height](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_height)[table.cell\_set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_color)[table.cell\_set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_halign)[table.cell\_set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_size)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[table.cell\_set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_width)[table.cell\_set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_tooltip)[table.cell\_set\_text\_formatting](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_formatting)

**table.cell\_set\_text\_color()**

The function sets the color of the text inside the cell.

SYNTAX

table.cell\_set\_text\_color(table\_id, column, row, text\_color) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**text\_color (series color)** The color of the text.

SEE ALSO

[table.cell\_set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_bgcolor)[table.cell\_set\_height](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_height)[table.cell\_set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text)[table.cell\_set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_halign)[table.cell\_set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_size)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[table.cell\_set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_width)[table.cell\_set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_tooltip)

**table.cell\_set\_text\_font\_family()**

The function sets the font family of the text inside the cell.

SYNTAX

table.cell\_set\_text\_font\_family(table\_id, column, row, text\_font\_family) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**text\_font\_family (series string)** The font family of the text. Possible values: [font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default), [font.family\_monospace](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_monospace).

EXAMPLE

//**@version=**6

indicator("Example of setting the table cell font")

var t = table.new(position.top\_left, rows = 1, columns = 1)

table.cell(t, 0, 0, "monospace", text\_color = color.blue)

table.cell\_set\_text\_font\_family(t, 0, 0, font.family\_monospace)

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[font.family\_default](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_default)[font.family\_monospace](https://www.tradingview.com/pine-script-reference/v6/#const_font.family_monospace)

**table.cell\_set\_text\_formatting()**

Sets the formatting attributes the drawing applies to displayed text.

SYNTAX

table.cell\_set\_text\_formatting(table\_id, column, row, text\_formatting) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**text\_formatting (const text\_format)** The formatting of the displayed text. Formatting options support addition. For example, text.format\_bold + text.format\_italic will make the text both bold and italicized. Possible values: [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none), [text.format\_bold](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_bold), [text.format\_italic](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_italic). Optional. The default is [text.format\_none](https://www.tradingview.com/pine-script-reference/v6/#var_text.format_none).

SEE ALSO

[table.cell\_set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_bgcolor)[table.cell\_set\_height](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_height)[table.cell\_set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_color)[table.cell\_set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_halign)[table.cell\_set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_size)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[table.cell\_set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_width)[table.cell\_set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_tooltip)[table.cell\_set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text)

**table.cell\_set\_text\_halign()**

The function sets the horizontal alignment of the cell's text.

SYNTAX

table.cell\_set\_text\_halign(table\_id, column, row, text\_halign) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**text\_halign (series string)** The horizontal alignment of a cell's text. Possible values: [text.align\_left](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_left), [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center), [text.align\_right](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_right).

SEE ALSO

[table.cell\_set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_bgcolor)[table.cell\_set\_height](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_height)[table.cell\_set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text)[table.cell\_set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_color)[table.cell\_set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_size)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[table.cell\_set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_width)[table.cell\_set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_tooltip)

**table.cell\_set\_text\_size()**

The function sets the size of the cell's text.

SYNTAX

table.cell\_set\_text\_size(table\_id, column, row, text\_size) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**text\_size (series int/string)** Size of the object. The size can be any positive integer, or one of the size.\* built-in constant strings. The constant strings and their equivalent integer values are: [size.auto](https://www.tradingview.com/pine-script-reference/v6/#const_size.auto) (0), [size.tiny](https://www.tradingview.com/pine-script-reference/v6/#const_size.tiny) (8), [size.small](https://www.tradingview.com/pine-script-reference/v6/#const_size.small) (10), [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal) (14), [size.large](https://www.tradingview.com/pine-script-reference/v6/#const_size.large) (20), [size.huge](https://www.tradingview.com/pine-script-reference/v6/#const_size.huge) (36). The default value is [size.normal](https://www.tradingview.com/pine-script-reference/v6/#const_size.normal) or 14.

SEE ALSO

[table.cell\_set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_bgcolor)[table.cell\_set\_height](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_height)[table.cell\_set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text)[table.cell\_set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_color)[table.cell\_set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_halign)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[table.cell\_set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_width)[table.cell\_set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_tooltip)

**table.cell\_set\_text\_valign()**

The function sets the vertical alignment of a cell's text.

SYNTAX

table.cell\_set\_text\_valign(table\_id, column, row, text\_valign) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**text\_valign (series string)** The vertical alignment of the cell's text. Possible values: [text.align\_top](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_top), [text.align\_center](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_center), [text.align\_bottom](https://www.tradingview.com/pine-script-reference/v6/#const_text.align_bottom).

SEE ALSO

[table.cell\_set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_bgcolor)[table.cell\_set\_height](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_height)[table.cell\_set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text)[table.cell\_set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_color)[table.cell\_set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_halign)[table.cell\_set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_size)[table.cell\_set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_width)[table.cell\_set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_tooltip)

**table.cell\_set\_tooltip()**

The function sets the tooltip in the specified cell.

SYNTAX

table.cell\_set\_tooltip(table\_id, column, row, tooltip) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**tooltip (series string)** The tooltip to be displayed inside the cell.

EXAMPLE

//**@version=**6

indicator("TABLE example")

var tLog = table.new(position = position.top\_left, rows = 1, columns = 2, bgcolor = color.yellow, border\_width=1)

table.cell(tLog, row = 0, column = 0, text = "sometext", text\_color = color.blue)

table.cell\_set\_tooltip(tLog, row = 0, column = 0, tooltip = "sometext")

SEE ALSO

[table.cell\_set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_bgcolor)[table.cell\_set\_height](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_height)[table.cell\_set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_color)[table.cell\_set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_halign)[table.cell\_set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_size)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[table.cell\_set\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_width)[table.cell\_set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text)

**table.cell\_set\_width()**

The function sets the width of the cell.

SYNTAX

table.cell\_set\_width(table\_id, column, row, width) → void

ARGUMENTS

**table\_id (series table)** A table object.

**column (series int)** The index of the cell's column. Numbering starts at 0.

**row (series int)** The index of the cell's row. Numbering starts at 0.

**width (series int/float)** The width of the cell as a % of the chart window. Passing 0 auto-adjusts the width based on the text inside of the cell.

SEE ALSO

[table.cell\_set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_bgcolor)[table.cell\_set\_height](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_height)[table.cell\_set\_text](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text)[table.cell\_set\_text\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_color)[table.cell\_set\_text\_halign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_halign)[table.cell\_set\_text\_size](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_size)[table.cell\_set\_text\_valign](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell_set_text_valign)[table.cell\_set\_tooltip](https://www.tradingview.com/pine-script-reference/v6/#fun_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.clear(table\_id, start\_column, start\_row, end\_column, end\_row) → void

ARGUMENTS

**table\_id (series table)** A table object.

**start\_column (series int)** The index of the column of the first cell to delete. Numbering starts at 0.

**start\_row (series int)** The index of the row of the first cell to delete. Numbering starts at 0.

**end\_column (series int)** The index of the column of the last cell to delete. Optional. The default is the argument used for start\_column. Numbering starts at 0.

**end\_row (series int)** The index of the row of the last cell to delete. Optional. The default is the argument used for start\_row. Numbering starts at 0.

EXAMPLE

//**@version=**6

indicator("A donut", overlay=true)

if barstate.islast

colNum = 8, rowNum = 8

padding = "◯"

donutTable = table.new(position.middle\_right, colNum, rowNum)

for c = 0 to colNum - 1

for r = 0 to rowNum - 1

table.cell(donutTable, c, r, text=padding, bgcolor=#face6e, text\_color=color.new(color.black, 100))

table.clear(donutTable, 2, 2, 5, 5)

SEE ALSO

[table.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_table.delete)[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)

**table.delete()**

The function deletes a table.

SYNTAX

table.delete(table\_id) → void

ARGUMENTS

**table\_id (series table)** A table object.

EXAMPLE

//**@version=**6

indicator("table.delete example")

var testTable = table.new(position = position.top\_right, columns = 2, rows = 1, bgcolor = color.yellow, border\_width = 1)

if barstate.islast

table.cell(table\_id = testTable, column = 0, row = 0, text = "Open is " + str.tostring(open))

table.cell(table\_id = testTable, column = 1, row = 0, text = "Close is " + str.tostring(close), bgcolor=color.teal)

if barstate.isrealtime

table.delete(testTable)

SEE ALSO

[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.clear](https://www.tradingview.com/pine-script-reference/v6/#fun_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.merge\_cells(table\_id, start\_column, start\_row, end\_column, end\_row) → void

ARGUMENTS

**table\_id (series table)** A table object.

**start\_column (series int)** The index of the column of the first cell to merge. Numbering starts at 0.

**start\_row (series int)** The index of the row of the first cell to merge. Numbering starts at 0.

**end\_column (series int)** The index of the column of the last cell to merge. Numbering starts at 0.

**end\_row (series int)** The index of the row of the last cell to merge. Numbering starts at 0.

EXAMPLE

//**@version=**6

indicator("table.merge\_cells example")

SMA50 = ta.sma(close, 50)

SMA100 = ta.sma(close, 100)

SMA200 = ta.sma(close, 200)

if barstate.islast

maTable = table.new(position.bottom\_right, 3, 3, bgcolor = color.gray, border\_width = 1, border\_color = color.black)

// Header

table.cell(maTable, 0, 0, text = "SMA Table")

table.merge\_cells(maTable, 0, 0, 2, 0)

// Cell Titles

table.cell(maTable, 0, 1, text = "SMA 50")

table.cell(maTable, 1, 1, text = "SMA 100")

table.cell(maTable, 2, 1, text = "SMA 200")

// Values

table.cell(maTable, 0, 2, bgcolor = color.white, text = str.tostring(SMA50))

table.cell(maTable, 1, 2, bgcolor = color.white, text = str.tostring(SMA100))

table.cell(maTable, 2, 2, bgcolor = color.white, text = str.tostring(SMA200))

REMARKS

This function will merge cells, even if their properties are not yet defined with [table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell).

The resulting merged cell inherits all of its values from the cell located at start\_column:start\_row, except width and height. The width and height of the resulting merged cell are based on the width/height of other cells in the neighboring columns/rows and cannot be set manually.

To modify the merged cell with any of the table.cell\_set\_\* functions, target the cell at the start\_column:start\_row coordinates.

An attempt to merge a cell that has already been merged will result in an error.

SEE ALSO

[table.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_table.delete)[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)

**table.new()**

The function creates a new table.

SYNTAX

table.new(position, columns, rows, bgcolor, frame\_color, frame\_width, border\_color, border\_width, force\_overlay) → series table

ARGUMENTS

**position (series string)** Position of the table. Possible values are: [position.top\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_left), [position.top\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_center), [position.top\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_right), [position.middle\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_left), [position.middle\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_center), [position.middle\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_right), [position.bottom\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_left), [position.bottom\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_center), [position.bottom\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_right).

**columns (series int)** The number of columns in the table.

**rows (series int)** The number of rows in the table.

**bgcolor (series color)** The background color of the table. Optional. The default is no color.

**frame\_color (series color)** The color of the outer frame of the table. Optional. The default is no color.

**frame\_width (series int)** The width of the outer frame of the table. Optional. The default is 0.

**border\_color (series color)** The color of the borders of the cells (excluding the outer frame). Optional. The default is no color.

**border\_width (series int)** The width of the borders of the cells (excluding the outer frame). Optional. The default is 0.

**force\_overlay (const bool)** If [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the drawing will display on the main chart pane, even when the script occupies a separate pane. Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("table.new example")

var testTable = table.new(position = position.top\_right, columns = 2, rows = 1, bgcolor = color.yellow, border\_width = 1)

if barstate.islast

table.cell(table\_id = testTable, column = 0, row = 0, text = "Open is " + str.tostring(open))

table.cell(table\_id = testTable, column = 1, row = 0, text = "Close is " + str.tostring(close), bgcolor=color.teal)

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](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell) and other table.cell\_\*() functions.

One [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new) call can only display one table (the last one drawn), but the function itself will be recalculated on each bar it is used on. For performance reasons, it is wise to use [table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new) in conjunction with either the [var](https://www.tradingview.com/pine-script-reference/v6/#kw_var) keyword (so the table object is only created on the first bar) or in an [if](https://www.tradingview.com/pine-script-reference/v6/#kw_if) [barstate.islast](https://www.tradingview.com/pine-script-reference/v6/#var_barstate.islast) block (so the table object is only created on the last bar).

SEE ALSO

[table.cell](https://www.tradingview.com/pine-script-reference/v6/#fun_table.cell)[table.clear](https://www.tradingview.com/pine-script-reference/v6/#fun_table.clear)[table.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_table.delete)[table.set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_bgcolor)[table.set\_border\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_color)[table.set\_border\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_width)[table.set\_frame\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_color)[table.set\_frame\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_width)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)

**table.set\_bgcolor()**

The function sets the background color of a table.

SYNTAX

table.set\_bgcolor(table\_id, bgcolor) → void

ARGUMENTS

**table\_id (series table)** A table object.

**bgcolor (series color)** The background color of the table. Optional. The default is no color.

SEE ALSO

[table.clear](https://www.tradingview.com/pine-script-reference/v6/#fun_table.clear)[table.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_table.delete)[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.set\_border\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_color)[table.set\_border\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_width)[table.set\_frame\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_color)[table.set\_frame\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_width)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_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.set\_border\_color(table\_id, border\_color) → void

ARGUMENTS

**table\_id (series table)** A table object.

**border\_color (series color)** The color of the borders. Optional. The default is no color.

SEE ALSO

[table.clear](https://www.tradingview.com/pine-script-reference/v6/#fun_table.clear)[table.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_table.delete)[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.set\_frame\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_color)[table.set\_border\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_width)[table.set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_bgcolor)[table.set\_frame\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_width)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_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.set\_border\_width(table\_id, border\_width) → void

ARGUMENTS

**table\_id (series table)** A table object.

**border\_width (series int)** The width of the borders. Optional. The default is 0.

SEE ALSO

[table.clear](https://www.tradingview.com/pine-script-reference/v6/#fun_table.clear)[table.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_table.delete)[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.set\_frame\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_color)[table.set\_frame\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_width)[table.set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_bgcolor)[table.set\_border\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_color)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)

**table.set\_frame\_color()**

The function sets the color of the outer frame of a table.

SYNTAX

table.set\_frame\_color(table\_id, frame\_color) → void

ARGUMENTS

**table\_id (series table)** A table object.

**frame\_color (series color)** The color of the frame of the table. Optional. The default is no color.

SEE ALSO

[table.clear](https://www.tradingview.com/pine-script-reference/v6/#fun_table.clear)[table.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_table.delete)[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.set\_border\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_color)[table.set\_border\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_width)[table.set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_bgcolor)[table.set\_frame\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_width)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)

**table.set\_frame\_width()**

The function set the width of the outer frame of a table.

SYNTAX

table.set\_frame\_width(table\_id, frame\_width) → void

ARGUMENTS

**table\_id (series table)** A table object.

**frame\_width (series int)** The width of the outer frame of the table. Optional. The default is 0.

SEE ALSO

[table.clear](https://www.tradingview.com/pine-script-reference/v6/#fun_table.clear)[table.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_table.delete)[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.set\_frame\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_color)[table.set\_border\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_width)[table.set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_bgcolor)[table.set\_border\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_color)[table.set\_position](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_position)

**table.set\_position()**

The function sets the position of a table.

SYNTAX

table.set\_position(table\_id, position) → void

ARGUMENTS

**table\_id (series table)** A table object.

**position (series string)** Position of the table. Possible values are: [position.top\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_left), [position.top\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_center), [position.top\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.top_right), [position.middle\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_left), [position.middle\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_center), [position.middle\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.middle_right), [position.bottom\_left](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_left), [position.bottom\_center](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_center), [position.bottom\_right](https://www.tradingview.com/pine-script-reference/v6/#const_position.bottom_right).

SEE ALSO

[table.clear](https://www.tradingview.com/pine-script-reference/v6/#fun_table.clear)[table.delete](https://www.tradingview.com/pine-script-reference/v6/#fun_table.delete)[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)[table.set\_bgcolor](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_bgcolor)[table.set\_border\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_color)[table.set\_border\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_border_width)[table.set\_frame\_color](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_color)[table.set\_frame\_width](https://www.tradingview.com/pine-script-reference/v6/#fun_table.set_frame_width)

**ticker.heikinashi()**2 overloads

Creates a ticker identifier for requesting Heikin Ashi bar values.

SYNTAX & OVERLOADS

[ticker.heikinashi(symbol) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.heikinashi-0)

[ticker.heikinashi(symbol) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.heikinashi-1)

ARGUMENTS

**symbol (simple string)** Symbol ticker identifier.

EXAMPLE

//**@version=**6

indicator("ticker.heikinashi", overlay=true)

heikinashi\_close = request.security(ticker.heikinashi(syminfo.tickerid), timeframe.period, close)

heikinashi\_aapl\_60\_close = request.security(ticker.heikinashi("AAPL"), "60", close)

plot(heikinashi\_close)

plot(heikinashi\_aapl\_60\_close)

RETURNS

String value of ticker id, that can be supplied to [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security) function.

SEE ALSO

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[ticker.renko](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.renko)[ticker.linebreak](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.linebreak)[ticker.kagi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.kagi)[ticker.pointfigure](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.pointfigure)

**ticker.inherit()**2 overloads

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

[ticker.inherit(from\_tickerid, symbol) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.inherit-0)

[ticker.inherit(from\_tickerid, symbol) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.inherit-1)

ARGUMENTS

**from\_tickerid (simple string)** The ticker ID to inherit modifiers from.

**symbol (simple string)** The symbol to construct the new ticker ID for.

EXAMPLE

//**@version=**6

indicator("ticker.inherit")

//**@variable** A "NASDAQ:AAPL" ticker ID with Extender Hours enabled.

tickerExtHours = ticker.new("NASDAQ", "AAPL", session.extended)

//**@variable** A Heikin Ashi ticker ID for "NASDAQ:AAPL" with Extended Hours enabled.

HAtickerExtHours = ticker.heikinashi(tickerExtHours)

//**@variable** The "NASDAQ:MSFT" symbol with no modifiers.

testSymbol = "NASDAQ:MSFT"

//**@variable** A ticker ID for "NASDAQ:MSFT" with inherited Heikin Ashi and Extended Hours modifiers.

testSymbolHAtickerExtHours = ticker.inherit(HAtickerExtHours, testSymbol)

//**@variable** The `close` price requested using "NASDAQ:MSFT" with inherited modifiers.

secData = request.security(testSymbolHAtickerExtHours, "60", close, ignore\_invalid\_symbol = true)

//**@variable** The `close` price requested using "NASDAQ:MSFT" without modifiers.

compareData = request.security(testSymbol, "60", close, ignore\_invalid\_symbol = true)

plot(secData, color = color.green)

plot(compareData)

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()**2 overloads

Creates a ticker identifier for requesting Kagi values.

SYNTAX & OVERLOADS

[ticker.kagi(symbol, reversal) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.kagi-0)

[ticker.kagi(symbol, reversal) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.kagi-1)

ARGUMENTS

**symbol (simple string)** Symbol ticker identifier.

**reversal (simple int/float)** Reversal amount (absolute price value).

EXAMPLE

//**@version=**6

indicator("ticker.kagi", overlay=true)

kagi\_tickerid = ticker.kagi(syminfo.tickerid, 3)

kagi\_close = request.security(kagi\_tickerid, timeframe.period, close)

plot(kagi\_close)

RETURNS

String value of ticker id, that can be supplied to [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security) function.

SEE ALSO

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[ticker.heikinashi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.heikinashi)[ticker.renko](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.renko)[ticker.linebreak](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.linebreak)[ticker.pointfigure](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.pointfigure)

**ticker.linebreak()**2 overloads

Creates a ticker identifier for requesting Line Break values.

SYNTAX & OVERLOADS

[ticker.linebreak(symbol, number\_of\_lines) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.linebreak-0)

[ticker.linebreak(symbol, number\_of\_lines) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.linebreak-1)

ARGUMENTS

**symbol (simple string)** Symbol ticker identifier.

**number\_of\_lines (simple int)** Number of line.

EXAMPLE

//**@version=**6

indicator("ticker.linebreak", overlay=true)

linebreak\_tickerid = ticker.linebreak(syminfo.tickerid, 3)

linebreak\_close = request.security(linebreak\_tickerid, timeframe.period, close)

plot(linebreak\_close)

RETURNS

String value of ticker id, that can be supplied to [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security) function.

SEE ALSO

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[ticker.heikinashi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.heikinashi)[ticker.renko](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.renko)[ticker.kagi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.kagi)[ticker.pointfigure](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.pointfigure)

**ticker.modify()**2 overloads

Creates a ticker identifier for requesting additional data for the script.

SYNTAX & OVERLOADS

[ticker.modify(tickerid, session, adjustment) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify-0)

[ticker.modify(tickerid, session, adjustment, backadjustment, settlement\_as\_close) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.modify-1)

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](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new) function.

**session (series string)** Session type. Optional argument. Possible values: [session.regular](https://www.tradingview.com/pine-script-reference/v6/#const_session.regular), [session.extended](https://www.tradingview.com/pine-script-reference/v6/#const_session.extended). Session type of the current chart is [syminfo.session](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.session). If session is not given, then [syminfo.session](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.session) value is used.

**adjustment (series string)** Adjustment type. Optional argument. Possible values: [adjustment.none](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.none), [adjustment.splits](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.splits), [adjustment.dividends](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.dividends). If adjustment is not given, then default adjustment value is used (can be different depending on particular instrument).

EXAMPLE

//**@version=**6

indicator("ticker\_modify", overlay=true)

t1 = ticker.new(syminfo.prefix, syminfo.ticker, session.regular, adjustment.splits)

c1 = request.security(t1, "D", close)

t2 = ticker.modify(t1, session.extended)

c2 = request.security(t2, "2D", close)

plot(c1)

plot(c2)

RETURNS

String value of ticker id, that can be supplied to [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security) function.

SEE ALSO

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.session](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.session)[session.extended](https://www.tradingview.com/pine-script-reference/v6/#const_session.extended)[session.regular](https://www.tradingview.com/pine-script-reference/v6/#const_session.regular)[ticker.heikinashi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.heikinashi)[adjustment.none](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.none)[adjustment.splits](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.splits)[adjustment.dividends](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.dividends)[backadjustment.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.inherit)[backadjustment.on](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.on)[backadjustment.off](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.off)[settlement\_as\_close.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.inherit)[settlement\_as\_close.on](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.on)[settlement\_as\_close.off](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.off)

**ticker.new()**2 overloads

Creates a ticker identifier for requesting additional data for the script.

SYNTAX & OVERLOADS

[ticker.new(prefix, ticker, session, adjustment) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new-0)

[ticker.new(prefix, ticker, session, adjustment, backadjustment, settlement\_as\_close) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new-1)

ARGUMENTS

**prefix (series string)** Exchange prefix. For example: 'BATS', 'NYSE', 'NASDAQ'. Exchange prefix of main series is [syminfo.prefix](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.prefix).

**ticker (series string)** Ticker name. For example 'AAPL', 'MSFT', 'EURUSD'. Ticker name of the main series is [syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker).

**session (series string)** Session type. Optional argument. Possible values: [session.regular](https://www.tradingview.com/pine-script-reference/v6/#const_session.regular), [session.extended](https://www.tradingview.com/pine-script-reference/v6/#const_session.extended). Session type of the current chart is [syminfo.session](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.session). If session is not given, then [syminfo.session](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.session) value is used.

**adjustment (series string)** Adjustment type. Optional argument. Possible values: [adjustment.none](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.none), [adjustment.splits](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.splits), [adjustment.dividends](https://www.tradingview.com/pine-script-reference/v6/#const_adjustment.dividends). If adjustment is not given, then default adjustment value is used (can be different depending on particular instrument).

EXAMPLE

//**@version=**6

indicator("ticker.new", overlay=true)

t = ticker.new(syminfo.prefix, syminfo.ticker, session.regular, adjustment.splits)

t2 = ticker.heikinashi(t)

c = request.security(t2, timeframe.period, low, barmerge.gaps\_on)

plot(c, style=plot.style\_linebr)

RETURNS

String value of ticker id, that can be supplied to [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security) function.

REMARKS

You may use return value of [ticker.new](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.new) function as input argument for [ticker.heikinashi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.heikinashi), [ticker.renko](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.renko), [ticker.linebreak](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.linebreak), [ticker.kagi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.kagi), [ticker.pointfigure](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.pointfigure) functions.

SEE ALSO

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[syminfo.session](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.session)[session.extended](https://www.tradingview.com/pine-script-reference/v6/#const_session.extended)[session.regular](https://www.tradingview.com/pine-script-reference/v6/#const_session.regular)[ticker.heikinashi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.heikinashi)[adjustment.none](https://www.tradingview.com/pine-script-reference/v6/#var_adjustment.none)[adjustment.splits](https://www.tradingview.com/pine-script-reference/v6/#var_adjustment.splits)[adjustment.dividends](https://www.tradingview.com/pine-script-reference/v6/#var_adjustment.dividends)[backadjustment.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.inherit)[backadjustment.on](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.on)[backadjustment.off](https://www.tradingview.com/pine-script-reference/v6/#const_backadjustment.off)[settlement\_as\_close.inherit](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.inherit)[settlement\_as\_close.on](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.on)[settlement\_as\_close.off](https://www.tradingview.com/pine-script-reference/v6/#const_settlement_as_close.off)

**ticker.pointfigure()**2 overloads

Creates a ticker identifier for requesting Point & Figure values.

SYNTAX & OVERLOADS

[ticker.pointfigure(symbol, source, style, param, reversal) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.pointfigure-0)

[ticker.pointfigure(symbol, source, style, param, reversal) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.pointfigure-1)

ARGUMENTS

**symbol (simple string)** Symbol ticker identifier.

**source (simple string)** The source for calculating Point & Figure. Possible values are: 'hl', 'close'.

**style (simple string)** Box Size Assignment Method: 'ATR', 'Traditional'.

**param (simple int/float)** ATR Length if style is equal to 'ATR', or Box Size if style is equal to 'Traditional'.

**reversal (simple int)** Reversal amount.

EXAMPLE

//**@version=**6

indicator("ticker.pointfigure", overlay=true)

pnf\_tickerid = ticker.pointfigure(syminfo.tickerid, "hl", "Traditional", 1, 3)

pnf\_close = request.security(pnf\_tickerid, timeframe.period, close)

plot(pnf\_close)

RETURNS

String value of ticker id, that can be supplied to [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security) function.

SEE ALSO

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[ticker.heikinashi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.heikinashi)[ticker.renko](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.renko)[ticker.linebreak](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.linebreak)[ticker.kagi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.kagi)

**ticker.renko()**2 overloads

Creates a ticker identifier for requesting Renko values.

SYNTAX & OVERLOADS

[ticker.renko(symbol, style, param, request\_wicks, source) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.renko-0)

[ticker.renko(symbol, style, param, request\_wicks, source) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.renko-1)

ARGUMENTS

**symbol (simple string)** Symbol ticker identifier.

**style (simple string)** Box Size Assignment Method: 'ATR', 'Traditional'.

**param (simple int/float)** ATR Length if style is equal to 'ATR', or Box Size if style is equal to 'Traditional'.

**request\_wicks (simple bool)** Specifies if wick values are returned for Renko bricks. When [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), [high](https://www.tradingview.com/pine-script-reference/v6/#var_high) and [low](https://www.tradingview.com/pine-script-reference/v6/#var_low) values requested from a symbol using the ticker formed by this function will include wick values when they are present. When [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), [high](https://www.tradingview.com/pine-script-reference/v6/#var_high) and [low](https://www.tradingview.com/pine-script-reference/v6/#var_low) will always be equal to either [open](https://www.tradingview.com/pine-script-reference/v6/#var_open) or [close](https://www.tradingview.com/pine-script-reference/v6/#var_close). Optional. The default is [false](https://www.tradingview.com/pine-script-reference/v6/#const_false). A detailed explanation of how Renko wicks are calculated can be found in our [Help Center](https://www.tradingview.com/support/solutions/43000481040-what-do-renko-wicks-mean/).

**source (simple string)** The source used to calculate bricks. Optional. Possible values: "Close", "OHLC". The default is "Close".

EXAMPLE

//**@version=**6

indicator("ticker.renko", overlay=true)

renko\_tickerid = ticker.renko(syminfo.tickerid, "ATR", 10)

renko\_close = request.security(renko\_tickerid, timeframe.period, close)

plot(renko\_close)

EXAMPLE

//**@version=**6

indicator("Renko candles", overlay=false)

renko\_tickerid = ticker.renko(syminfo.tickerid, "ATR", 10)

[renko\_open, renko\_high, renko\_low, renko\_close] = request.security(renko\_tickerid, timeframe.period, [open, high, low, close])

plotcandle(renko\_open, renko\_high, renko\_low, renko\_close, color = renko\_close > renko\_open ? color.green : color.red)

RETURNS

String value of ticker id, that can be supplied to [request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security) function.

SEE ALSO

[syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid)[syminfo.ticker](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.ticker)[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)[ticker.heikinashi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.heikinashi)[ticker.linebreak](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.linebreak)[ticker.kagi](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.kagi)[ticker.pointfigure](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.pointfigure)

**ticker.standard()**2 overloads

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

[ticker.standard(symbol) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.standard-0)

[ticker.standard(symbol) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_ticker.standard-1)

ARGUMENTS

**symbol (simple string)** A ticker ID to be converted into its standard form. Optional. The default is [syminfo.tickerid](https://www.tradingview.com/pine-script-reference/v6/#var_syminfo.tickerid).

EXAMPLE

//**@version=**6

indicator("ticker.standard", overlay = true)

// This script should be run on a non-standard chart such as HA, Renko...

// Requests data from the chart type the script is running on.

chartTypeValue = request.security(syminfo.tickerid, "1D", close)

// Request data from the standard chart type, regardless of the chart type the script is running on.

standardChartValue = request.security(ticker.standard(syminfo.tickerid), "1D", close)

// This will not use a standard ticker ID because the `symbol` argument contains only the ticker — not the prefix (exchange).

standardChartValue2 = request.security(ticker.standard(syminfo.ticker), "1D", close)

plot(chartTypeValue)

plot(standardChartValue, color = color.green)

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

[request.security](https://www.tradingview.com/pine-script-reference/v6/#fun_request.security)

**time()**3 overloads

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

[time(timeframe, bars\_back) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_time-0)

[time(timeframe, session, bars\_back) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_time-1)

[time(timeframe, session, timezone, bars\_back) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_time-2)

ARGUMENTS

**timeframe (series string)** Timeframe. An empty string is interpreted as the current timeframe of the chart.

**bars\_back (series int)** Optional. The bar offset on the script's main timeframe. If the value is positive, the function retrieves the timestamp of the bar N bars back relative to the current bar on the main timeframe. If the value is a negative number from -1 to -500, the function retrieves the expected time of a future bar on that timeframe. The default is 0.

EXAMPLE

//**@version=**6

indicator("Time", overlay=true)

// Try this on chart AAPL,1

timeinrange(res, sess) => not na(time(res, sess, "America/New\_York")) ? 1 : 0

plot(timeinrange("1", "1300-1400"), color=color.red)

// This plots 1.0 at every start of 10 minute bar on a 1 minute chart:

newbar(res) => ta.change(time(res)) == 0 ? 0 : 1

plot(newbar("10"))

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.

If the days aren't specified, the session is considered to have been set from Sunday (1) to Saturday (7), i.e. "1100-2000" is the same as "1100-1200:1234567".

You can change that by specifying the days. For example, on a symbol that is traded seven days a week with the 24-hour trading session the following script will not color Saturdays and Sundays:

EXAMPLE

//**@version=**6

indicator("Time", overlay=true)

t1 = time(timeframe.period, "0000-0000:23456")

bgcolor(not na(t1) ? color.new(color.blue, 90) : na)

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

//**@version=**6

indicator("Time", overlay=true)

t1 = time(timeframe.period, "1000-1100,1400-1500:23456")

bgcolor(not na(t1) ? color.new(color.blue, 90) : na)

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](https://www.tradingview.com/pine-script-reference/v6/#var_time)

**time\_close()**3 overloads

Returns the UNIX time of the current bar's close for the specified timeframe and session, or [na](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#var_na) on the chart's realtime bars.

SYNTAX & OVERLOADS

[time\_close(timeframe, bars\_back) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_time_close-0)

[time\_close(timeframe, session, bars\_back) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_time_close-1)

[time\_close(timeframe, session, timezone, bars\_back) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_time_close-2)

ARGUMENTS

**timeframe (series string)** Resolution. An empty string is interpreted as the current resolution of the chart.

**bars\_back (series int)** Optional. The bar offset on the script's main timeframe. If the value is positive, the function retrieves the timestamp of the bar N bars back relative to the current bar on the main timeframe. If the value is a negative number from -1 to -500, the function retrieves the expected time of a future bar on that timeframe. The default is 0.

EXAMPLE

//**@version=**6

indicator("Time", overlay=true)

t1 = time\_close(timeframe.period, "1200-1300", "America/New\_York")

bgcolor(not na(t1) ? color.new(color.blue, 90) : na)

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\_close](https://www.tradingview.com/pine-script-reference/v6/#var_time_close)

**timeframe.change()**

Detects changes in the specified timeframe.

SYNTAX

timeframe.change(timeframe) → series bool

ARGUMENTS

**timeframe (series string)** String formatted according to the [User manual's timeframe string specifications](https://www.tradingview.com/pine-script-docs/concepts/timeframes/#timeframe-string-specifications).

EXAMPLE

//**@version=**6

// Run this script on an intraday chart.

indicator("New day started", overlay = true)

// Highlights the first bar of the new day.

isNewDay = timeframe.change("1D")

bgcolor(isNewDay ? color.new(color.green, 80) : na)

RETURNS

Returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) on the first bar of a new timeframe, [false](https://www.tradingview.com/pine-script-reference/v6/#const_false) otherwise.

**timeframe.from\_seconds()**2 overloads

Converts a number of seconds into a valid timeframe string.

SYNTAX & OVERLOADS

[timeframe.from\_seconds(seconds) → simple string](https://www.tradingview.com/pine-script-reference/v6/#fun_timeframe.from_seconds-0)

[timeframe.from\_seconds(seconds) → series string](https://www.tradingview.com/pine-script-reference/v6/#fun_timeframe.from_seconds-1)

ARGUMENTS

**seconds (simple int)** The number of seconds in the timeframe.

EXAMPLE

//**@version=**6

indicator("HTF Close", "", true)

**int** chartTf = timeframe.in\_seconds()

**string** tfTimes5 = timeframe.from\_seconds(chartTf \* 5)

**float** htfClose = request.security(syminfo.tickerid, tfTimes5, close)

plot(htfClose)

RETURNS

A timeframe string compliant with [timeframe string specifications](https://www.tradingview.com/pine-script-docs/concepts/timeframes/#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".

If the seconds exactly represent two or more valid timeframes, the one with the larger base unit will be used. Thus 604,800 seconds (7 days) returns "1W", not "7D".

All values above 31,622,400 (366 days) return "12M".

SEE ALSO

[timeframe.in\_seconds](https://www.tradingview.com/pine-script-reference/v6/#fun_timeframe.in_seconds)[request.security](https://www.tradingview.com/pine-script-reference/v6/#var_request.security)[request.security\_lower\_tf](https://www.tradingview.com/pine-script-reference/v6/#var_request.security_lower_tf)

**timeframe.in\_seconds()**2 overloads

Converts a timeframe string into seconds.

SYNTAX & OVERLOADS

[timeframe.in\_seconds(timeframe) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_timeframe.in_seconds-0)

[timeframe.in\_seconds(timeframe) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_timeframe.in_seconds-1)

ARGUMENTS

**timeframe (simple string)** Timeframe string in [timeframe string specifications](https://www.tradingview.com/pine-script-docs/concepts/timeframes/#timeframe-string-specifications) format. Optional. The default is [timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period).

EXAMPLE

//**@version=**6

indicator("`timeframe\_in\_seconds()`")

// Get a user-selected timeframe.

tfInput = input.timeframe("1D")

// Convert it into an "int" number of seconds.

secondsInTf = timeframe.in\_seconds(tfInput)

plot(secondsInTf)

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

[input.timeframe](https://www.tradingview.com/pine-script-reference/v6/#fun_input.timeframe)[timeframe.period](https://www.tradingview.com/pine-script-reference/v6/#var_timeframe.period)[timeframe.from\_seconds](https://www.tradingview.com/pine-script-reference/v6/#fun_timeframe.from_seconds)

**timestamp()**5 overloads

Function timestamp returns UNIX time of specified date and time.

SYNTAX & OVERLOADS

[timestamp(dateString) → const int](https://www.tradingview.com/pine-script-reference/v6/#fun_timestamp-0)

[timestamp(year, month, day, hour, minute, second) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_timestamp-1)

[timestamp(year, month, day, hour, minute, second) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_timestamp-2)

[timestamp(timezone, year, month, day, hour, minute, second) → simple int](https://www.tradingview.com/pine-script-reference/v6/#fun_timestamp-3)

[timestamp(timezone, year, month, day, hour, minute, second) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_timestamp-4)

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](https://tools.ietf.org/html/rfc2822#section-3.3) or [ISO 8601](https://en.wikipedia.org/wiki/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

//**@version=**6

indicator("timestamp")

plot(timestamp(2016, 01, 19, 09, 30), linewidth=3, color=color.green)

plot(timestamp(syminfo.timezone, 2016, 01, 19, 09, 30), color=color.blue)

plot(timestamp(2016, 01, 19, 09, 30), color=color.yellow)

plot(timestamp("GMT+6", 2016, 01, 19, 09, 30))

plot(timestamp(2019, 06, 19, 09, 30, 15), color=color.lime)

plot(timestamp("GMT+3", 2019, 06, 19, 09, 30, 15), color=color.fuchsia)

plot(timestamp("Feb 01 2020 22:10:05"))

plot(timestamp("2011-10-10T14:48:00"))

plot(timestamp("04 Dec 1995 00:12:00 GMT+5"))

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](https://www.tradingview.com/pine-script-reference/v6/#fun_time)

**weekofyear()**2 overloads

SYNTAX & OVERLOADS

[weekofyear(time) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_weekofyear-0)

[weekofyear(time, timezone) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_weekofyear-1)

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.

Note that this function 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

[weekofyear](https://www.tradingview.com/pine-script-reference/v6/#var_weekofyear)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[year](https://www.tradingview.com/pine-script-reference/v6/#fun_year)[month](https://www.tradingview.com/pine-script-reference/v6/#fun_month)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#fun_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#fun_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#fun_second)

**year()**2 overloads

SYNTAX & OVERLOADS

[year(time) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_year-0)

[year(time, timezone) → series int](https://www.tradingview.com/pine-script-reference/v6/#fun_year-1)

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.

Note that this function 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 UTC-4) this value can be lower by 1 than the year of the trading day.

SEE ALSO

[year](https://www.tradingview.com/pine-script-reference/v6/#var_year)[time](https://www.tradingview.com/pine-script-reference/v6/#fun_time)[month](https://www.tradingview.com/pine-script-reference/v6/#fun_month)[dayofmonth](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofmonth)[dayofweek](https://www.tradingview.com/pine-script-reference/v6/#fun_dayofweek)[hour](https://www.tradingview.com/pine-script-reference/v6/#fun_hour)[minute](https://www.tradingview.com/pine-script-reference/v6/#fun_minute)[second](https://www.tradingview.com/pine-script-reference/v6/#fun_second)

## **Keywords**

**and**

Logical AND. Applicable to boolean expressions.

SYNTAX

expr1 and expr2

RETURNS

Boolean value, or series of boolean values.

REMARKS

If expr1 evaluates to [false](https://www.tradingview.com/pine-script-reference/v6/#const_false), the and operator returns [false](https://www.tradingview.com/pine-script-reference/v6/#const_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.

Each field in an enum has a const string title. Scripts can access the fields in an enum using dot notation, similar to accessing the fields of a user-defined type.

Each field represents a value of the enumName enum. Scripts can declare each field in an enum with an optional const string title. If a field's title is not specified, its title is the string representation of its name. Use [str.tostring](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tostring) on an enum field to retrieve its title.

SYNTAX

[export ]enum <enumName>

<field\_1> [= <title\_1>]

<field\_2> [= <title\_2>]

...

<field\_N> [= <title\_N>]

One can use an enum to quickly create a dropdown input with the help of the [input.enum](https://www.tradingview.com/pine-script-reference/v6/#fun_input.enum) function. The options that appear in the dropdown represent the titles of the enum fields.

EXAMPLE

//**@version=**6

indicator("Session highlight", overlay = true)

//**@enum** Contains fields with popular timezones as titles.

//**@field** exch Has an empty string as the title to represent the chart timezone.

enum tz

utc = "UTC"

exch = ""

ny = "America/New\_York"

chi = "America/Chicago"

lon = "Europe/London"

tok = "Asia/Tokyo"

//**@variable** The session string.

selectedSession = input.session("1200-1500", "Session")

//**@variable** The selected timezone. The input's dropdown contains the fields in the `tz` enum.

selectedTimezone = input.enum(tz.utc, "Session Timezone")

//**@variable** Is `true` if the current bar's time is in the specified session.

**bool** inSession = false

if not na(time("", selectedSession, str.tostring(selectedTimezone)))

inSession := true

// Highlight the background when `inSession` is `true`.

bgcolor(inSession ? color.new(color.green, 90) : na, title = "Active session highlight")

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

//**@version=**6

indicator("Map with enum keys")

//**@enum** Contains fields with titles representing ticker IDs.

//**@field** aapl Has an Apple ticker ID as its title.

//**@field** tsla Has a Tesla ticker ID as its title.

//**@field** amzn Has an Amazon ticker ID as its title.

enum symbols

aapl = "NASDAQ:AAPL"

tsla = "NASDAQ:TSLA"

amzn = "NASDAQ:AMZN"

//**@variable** A map that accepts fields from the `symbols` enum as keys and "float" values.

**map**<**symbols**, **float**> data = map.new<**symbols**, **float**>()

// Put key-value pairs into the `data` map.

data.put(symbols.aapl, request.security(str.tostring(symbols.aapl), timeframe.period, close))

data.put(symbols.tsla, request.security(str.tostring(symbols.tsla), timeframe.period, close))

data.put(symbols.amzn, request.security(str.tostring(symbols.amzn), timeframe.period, close))

// Plot the value from the `data` map accessed by the `symbols.aapl` key.

plot(data.get(symbols.aapl))

**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

//**@version=**6

//**@description** Library of debugging functions.

library("Debugging\_library", overlay = true)

//**@function** Displays a string as a table cell for debugging purposes.

//**@param** txt String to display.

//**@returns** Void.

export print(**string** txt) =>

var **table** t = table.new(position.middle\_right, 1, 1)

table.cell(t, 0, 0, txt, bgcolor = color.yellow)

// Using the function from inside the library to show an example on the published chart.

// This has no impact on scripts using the library.

print("Library Test")

REMARKS

Each library must have at least one exported function or user-defined type (UDT).

Exported functions cannot use variables from the global scope if they are arrays, mutable variables (reassigned with :=), or variables of 'input' form.

Exported functions cannot use request.\*() functions.

Exported functions must explicitly declare each parameter's type and all parameters must be used in the function's body. By default, all arguments passed to exported functions are of the [series](https://www.tradingview.com/pine-script-reference/v6/#type_series) form, unless they are explicitly specified as [simple](https://www.tradingview.com/pine-script-reference/v6/#type_simple) in the function's signature.

The @description, @function, @param, @type, @field, and @returns compiler annotations are used to automatically generate the library's description and release notes, and in the Pine Script® Editor's tooltips.

SEE ALSO

[library](https://www.tradingview.com/pine-script-reference/v6/#fun_library)[import](https://www.tradingview.com/pine-script-reference/v6/#kw_import)[simple](https://www.tradingview.com/pine-script-reference/v6/#type_simple)[series](https://www.tradingview.com/pine-script-reference/v6/#type_series)[type](https://www.tradingview.com/pine-script-reference/v6/#kw_type)

**for**

The 'for' structure allows the repeated execution of a number of statements:

SYNTAX

[var\_declaration =] for counter = from\_num to to\_num [by step\_num]

statements | continue | break

return\_expression

**var\_declaration** - An optional variable declaration that will be assigned the value of the loop's return\_expression.

**counter** - A variable holding the value of the loop's counter, which is incremented/decremented by 1 or by the step\_num value on each iteration of the loop.

**from\_num** - The starting value of the counter. "series int/float" values/expressions are allowed.

**to\_num** - The end value of the counter. When the counter becomes greater than to\_num (or less than to\_num in cases where from\_num > to\_num) the loop is broken. "series int/float" values/expressions are allowed, but they are evaluated only on the loop's first iteration.

**step\_num** - The increment/decrement value of the counter. It is optional. The default value is +1 or -1, depending on which of from\_num or to\_num is the greatest. When a value is used, the counter is also incremented/decremented depending on which of from\_num or to\_num is the greatest, so the +/- sign of step\_num is optional.

**statements | continue | break** - Any number of statements, or the 'continue' or 'break' keywords, indented by 4 spaces or a tab.

**return\_expression** - The loop's return value which is assigned to the variable in var\_declaration if one is present. If the loop exits because of a 'continue' or 'break' keyword, the loop's return value is that of the last variable assigned a value before the loop's exit.

**continue** - A keyword that can only be used in loops. It causes the next iteration of the loop to be executed.

**break** - A keyword that exits the loop.

EXAMPLE

//**@version=**6

indicator("for")

// Here, we count the quantity of bars in a given 'lookback' length which closed above the current bar's close

qtyOfHigherCloses(lookback) =>

**int** result = 0

for i = 1 to lookback

if close[i] > close

result += 1

result

plot(qtyOfHigherCloses(14))

EXAMPLE

//**@version=**6

indicator("`for` loop with a step")

a = array.from(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

sum = 0.0

for i = 0 to 9 by 5

// Because the step is set to 5, we are adding only the first (0) and the sixth (5) value from the array `a`.

sum += array.get(a, i)

plot(sum)

SEE ALSO

[for...in](https://www.tradingview.com/pine-script-reference/v6/#kw_for...in)[while](https://www.tradingview.com/pine-script-reference/v6/#kw_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 =] for array\_element in array\_id

statements | continue | break

return\_expression

[var\_declaration =] for [index, array\_element] in array\_id

statements | continue | break

return\_expression

**var\_declaration** - An optional variable declaration that will be assigned the value of the loop's return\_expression.

**index** - An optional variable that tracks the current iteration's index. Indexing starts at 0. The variable is immutable in the loop's body. When used, it must be included in a tuple also containing array\_element.

**array\_element** - A variable containing each successive array element to be processed in the loop. The variable is immutable in the loop's body.

**array\_id** - The ID of the array over which the loop is iterated.

**statements | continue | break** - Any number of statements, or the 'continue' or 'break' keywords, indented by 4 spaces or a tab.

**return\_expression** - The loop's return value assigned to the variable in var\_declaration, if one is present. If the loop exits because of a 'continue' or 'break' keyword, the loop's return value is that of the last variable assigned a value before the loop's exit.

**continue** - A keyword that can only be used in loops. It causes the next iteration of the loop to be executed.

**break** - A keyword that exits the loop.

Scripts can modify arrays and matrices while iterating over their elements with this structure. However, maps cannot change while looping through their key-value pairs. To modify a map within a for...in loop, iterate over the key-value pairs of a copy or over the elements in its [map.keys](https://www.tradingview.com/pine-script-reference/v6/#fun_map.keys) array.

Here, we use the single-argument form of for...in to determine on each bar how many of the bar's OHLC values are greater than the SMA of 'close' values:

EXAMPLE

//**@version=**6

indicator("for...in")

// Here we determine on each bar how many of the bar's OHLC values are greater than the SMA of 'close' values

**float**[] ohlcValues = array.from(open, high, low, close)

qtyGreaterThan(value, array) =>

**int** result = 0

for currentElement in **array**

if currentElement > value

result += 1

result

plot(qtyGreaterThan(ta.sma(close, 20), ohlcValues))

Here, we use the two-argument form of [for...in](https://www.tradingview.com/pine-script-reference/v6/#kw_for...in) to set the values of our isPos array to true when their corresponding value in our valuesArray array is positive:

EXAMPLE

//**@version=**6

indicator("for...in")

var valuesArray = array.from(4, -8, 11, 78, -16, 34, 7, 99, 0, 55)

var isPos = array.new\_bool(10, false)

for [index, value] in valuesArray

if value > 0

array.set(isPos, index, true)

if barstate.islastconfirmedhistory

label.new(bar\_index, high, str.tostring(isPos))

Iterate through matrix rows as arrays.

EXAMPLE

//**@version=**6

indicator("`for ... in` matrix Example")

// Create a 2x3 matrix with values `4`.

matrix1 = matrix.new<**int**>(2, 3, 4)

sum = 0.0

// Loop through every row of the matrix.

for rowArray in matrix1

// Sum values of the every row

sum += array.sum(rowArray)

plot(sum)

SEE ALSO

[for](https://www.tradingview.com/pine-script-reference/v6/#kw_for)[while](https://www.tradingview.com/pine-script-reference/v6/#kw_while)[array.sum](https://www.tradingview.com/pine-script-reference/v6/#fun_array.sum)[array.min](https://www.tradingview.com/pine-script-reference/v6/#fun_array.min)[array.max](https://www.tradingview.com/pine-script-reference/v6/#fun_array.max)

**if**

If statement defines what block of statements must be executed when conditions of the expression are satisfied.

To have access to and use the if statement, one should specify the version >= 2 of Pine Script® language in the very first line of code, for example: //@version=6

The 4th version of Pine Script® Language allows you to use “else if” syntax.

General code form:

SYNTAX

var\_declarationX = if condition

var\_decl\_then0

var\_decl\_then1

…

var\_decl\_thenN

else if [optional block]

var\_decl\_else0

var\_decl\_else1

…

var\_decl\_elseN

else

var\_decl\_else0

var\_decl\_else1

…

var\_decl\_elseN

return\_expression\_else

where

**var\_declarationX** — this variable gets the value of the if statement

**condition** — if the condition is true, the logic from the block 'then' (var\_decl\_then0, var\_decl\_then1, etc.) is used.

If the condition is false, the logic from the block 'else' (var\_decl\_else0, var\_decl\_else1, etc.) is used.

**return\_expression\_then**, **return\_expression\_else** — the last expression from the block then or from the block else will return the final value of the statement. If declaration of the variable is in the end, its value will be the result.

The type of returning value of the if statement depends on return\_expression\_then and return\_expression\_else type (their types must match: it is not possible to return an integer value from then, while you have a string value in else block).

EXAMPLE

//**@version=**6

indicator("if")

// This code compiles

x = if close > open

close

else

open

// This code doesn’t compile

// y = if close > open

// close

// else

// "open"

plot(x)

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

//**@version=**6

indicator("if")

x = if close > open

close

// If current close > current open, then x = close.

// Otherwise the x = na.

plot(x)

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

//**@version=**6

indicator("if")

x = if open > close

5

else if high > low

close

else

open

plot(x)

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

//**@version=**6

strategy("if")

if (ta.crossover(high, low))

strategy.entry("BBandLE", strategy.long, stop=low, oca\_name="BollingerBands", oca\_type=strategy.oca.cancel, comment="BBandLE")

else

strategy.cancel(id="BBandLE")

If statements can include each other:

EXAMPLE

//**@version=**6

indicator("if")

**float** x = na

if close > open

if close > close[1]

x := close

else

x := close[1]

else

x := open

plot(x)

**import**

Used to load an external [library](https://www.tradingview.com/pine-script-reference/v6/#fun_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

import {username}/{libraryName}/{libraryVersion} as {alias}

ARGUMENTS

**username (literal string)** User name of the library's author.

**libraryName (literal string)** Name of the imported library, which corresponds to the title argument used by the author in his library script.

**libraryVersion (literal int)** Version number of the imported library.

**alias (literal string)** A non-numeric identifier used as a namespace to refer to the library's functions. Optional. The default is the libraryName string.

EXAMPLE

//**@version=**6

indicator("num\_methods import")

// Import the first version of the username’s "num\_methods" library and assign it to the "m" namespace",

import username/num\_methods/1 as m

// Call the “sinh()” function from the imported library

y = m.sinh(3.14)

// Plot value returned by the "sinh()" function",

plot(y)

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

[library](https://www.tradingview.com/pine-script-reference/v6/#fun_library)[export](https://www.tradingview.com/pine-script-reference/v6/#kw_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.

The first parameter of a method declaration must be explicitly typified.

SYNTAX

[export] method <functionName>(<paramType> <paramName> [= <defaultValue>], …) =>

<functionBlock>

EXAMPLE

//**@version=**6

indicator("")

var prices = array.new<**float**>()

//**@function** Pushes a new value into the array and removes the first one if the resulting array is greater than `maxSize`. Can be used as a method.

method maintainArray(**array**<**float**> id, maxSize, value) =>

id.push(value)

if id.size() > maxSize

id.shift()

prices.maintainArray(50, close)

// The method can also be called like a function, without using dot notation.

// In this case an argument must be supplied for its first parameter.

// maintainArray(prices, 50, close)

// This calls the `array.avg()` built-in using dot notation with the `prices` array.

// It is possible because built-in functions belonging to some namespaces that are a special Pine type

// can be invoked with method notation when the function's first parameter is an ID of that type.

// Those namespaces are: `array`, `matrix`, `line`, `linefill`, `label`, `box`, and `table`.

plot(prices.avg())

**not**

Logical negation (NOT). Applicable to boolean expressions.

SYNTAX

not expr1

RETURNS

Boolean value, or series of boolean values.

**or**

Logical OR. Applicable to boolean expressions.

SYNTAX

expr1 or expr2

RETURNS

Boolean value, or series of boolean values.

REMARKS

If expr1 evaluates to [true](https://www.tradingview.com/pine-script-reference/v6/#const_true), the or operator returns [true](https://www.tradingview.com/pine-script-reference/v6/#const_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

[variable\_declaration = ] switch expression

value1 => local\_block

value2 => local\_block

…

=> default\_local\_block

[variable\_declaration = ] switch

condition1 => local\_block

condition2 => local\_block

…

=> default\_local\_block

Switch with an expression:

EXAMPLE

//**@version=**6

indicator("Switch using an expression")

**string** i\_maType = input.string("EMA", "MA type", options = ["EMA", "SMA", "RMA", "WMA"])

**float** ma = switch i\_maType

"EMA" => ta.ema(close, 10)

"SMA" => ta.sma(close, 10)

"RMA" => ta.rma(close, 10)

// Default used when the three first cases do not match.

=> ta.wma(close, 10)

plot(ma)

Switch without an expression:

EXAMPLE

//**@version=**6

strategy("Switch without an expression", overlay = true)

**bool** longCondition = ta.crossover( ta.sma(close, 14), ta.sma(close, 28))

**bool** shortCondition = ta.crossunder(ta.sma(close, 14), ta.sma(close, 28))

switch

longCondition => strategy.entry("Long ID", strategy.long)

shortCondition => strategy.entry("Short ID", strategy.short)

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

[if](https://www.tradingview.com/pine-script-reference/v6/#kw_if)[?:](https://www.tradingview.com/pine-script-reference/v6/#op_?:)

**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

[export ]type <UDT\_identifier>

[varip ]<field\_type> <field\_name> [= <value>]

…

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](https://www.tradingview.com/pine-script-reference/v6/#var_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](https://www.tradingview.com/pine-script-reference/v6/#const_true).

Field declarations can include the [varip](https://www.tradingview.com/pine-script-reference/v6/#kw_varip) keyword, in which case the field values persist between successive script iterations on the same bar.

For more information see the User Manual's sections on [defining UDTs](https://www.tradingview.com/pine-script-docs/language/type-system#user-defined-types) and [using objects](https://www.tradingview.com/pine-script-docs/language/objects/).

Libraries can export UDTs. See the [Libraries](https://www.tradingview.com/pine-script-docs/concepts/libraries/#user-defined-types-and-objects) page of our User Manual to learn more.

EXAMPLE

//**@version=**6

indicator("Multi Time Period Chart", overlay = true)

timeframeInput = input.timeframe("1D")

type bar

**float** o = open

**float** h = high

**float** l = low

**float** c = close

**int** t = time

drawBox(**bar** b, right) =>

**bar** s = bar.new()

**color** boxColor = b.c >= b.o ? color.green : color.red

box.new(b.t, b.h, right, b.l, boxColor, xloc = xloc.bar\_time, bgcolor = color.new(boxColor, 90))

updateBox(**box** boxId, **bar** b) =>

**color** boxColor = b.c >= b.o ? color.green : color.red

box.set\_border\_color(boxId, boxColor)

box.set\_bgcolor(boxId, color.new(boxColor, 90))

box.set\_top(boxId, b.h)

box.set\_bottom(boxId, b.l)

box.set\_right(boxId, time)

secBar = request.security(syminfo.tickerid, timeframeInput, bar.new())

if not na(secBar)

// To avoid a runtime error, only process data when an object exists.

if not barstate.islast

if timeframe.change(timeframeInput)

// On historical bars, draw a new box in the past when the HTF closes.

drawBox(secBar, time[1])

else

var **box** lastBox = na

if na(lastBox) or timeframe.change(timeframeInput)

// On the last bar, only draw a new current box the first time we get there or when HTF changes.

lastBox := drawBox(secBar, time)

else

// On other chart updates, use setters to modify the current box.

updateBox(lastBox, secBar)

**var**

**var** is the keyword used for assigning and one-time initializing of the variable.

Normally, a syntax of assignment of variables, which doesn’t include the keyword var, results in the value of the variable being overwritten with every update of the data. Contrary to that, when assigning variables with the keyword var, they can “keep the state” despite the data updating, only changing it when conditions within if-expressions are met.

SYNTAX

var variable\_name = expression

where:

**variable\_name** - any name of the user’s variable that’s allowed in Pine Script® (can contain capital and lowercase Latin characters, numbers, and underscores (\_), but can’t start with a number).

**expression** - any arithmetic expression, just as with defining a regular variable. The expression will be calculated and assigned to a variable once.

EXAMPLE

//**@version=**6

indicator("Var keyword example")

var a = close

var b = 0.0

var c = 0.0

var green\_bars\_count = 0

if close > open

var x = close

b := x

green\_bars\_count := green\_bars\_count + 1

if green\_bars\_count >= 10

var y = close

c := y

plot(a)

plot(b)

plot(c)

The variable 'a' keeps the closing price of the first bar for each bar in the series.

The variable 'b' keeps the closing price of the first "green" bar in the series.

The variable 'c' keeps the closing price of the tenth "green" 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](https://www.tradingview.com/pine-script-reference/v6/#kw_type). It’s similar to the [var](https://www.tradingview.com/pine-script-reference/v6/#kw_var) keyword, but variables and fields declared with [varip](https://www.tradingview.com/pine-script-reference/v6/#kw_varip) retain their values between executions of the script on the same bar.

SYNTAX

varip [<variable\_type> ]<variable\_name> = <expression>

[export ]type <UDT\_identifier>

varip <field\_type> <field\_name> [= <value>]

where:

**variable\_type** - An optional fundamental type ([int](https://www.tradingview.com/pine-script-reference/v6/#type_int), [float](https://www.tradingview.com/pine-script-reference/v6/#type_float), [bool](https://www.tradingview.com/pine-script-reference/v6/#type_bool), [color](https://www.tradingview.com/pine-script-reference/v6/#type_color), [string](https://www.tradingview.com/pine-script-reference/v6/#type_string)) or a user-defined type, or an array or matrix of one of those types. Special types are not compatible with this keyword.

**variable\_name** - A [valid identifier](https://www.tradingview.com/pine-script-docs/language/identifiers/). The variable can also be an object created from a UDT.

**expression** - Any arithmetic expression, just as when defining a regular variable. The expression will be calculated and assigned to the variable only once, on the first bar.

**UDT\_identifier, field\_type, field\_name, value** - Constructs related to user-defined types as described in the [type](https://www.tradingview.com/pine-script-reference/v6/#kw_type) section.

EXAMPLE

//**@version=**6

indicator("varip")

varip **int** v = -1

v := v + 1

plot(v)

With [var](https://www.tradingview.com/pine-script-reference/v6/#kw_var), v would equal the value of the [bar\_index](https://www.tradingview.com/pine-script-reference/v6/#var_bar_index). On historical bars, where the script calculates only once per chart bar, the value of v is the same as with [var](https://www.tradingview.com/pine-script-reference/v6/#kw_var). However, on realtime bars, the script will evaluate the expression on each new chart update, producing a different result.

EXAMPLE

//**@version=**6

indicator("varip with types")

type barData

**int** index = -1

varip **int** ticks = -1

var currBar = barData.new()

currBar.index += 1

currBar.ticks += 1

// Will be equal to bar\_index on all bars

plot(currBar.index)

// In real time, will increment per every tick on the chart

plot(currBar.ticks)

The same [+=](https://www.tradingview.com/pine-script-reference/v6/#op_+=) 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](https://www.tradingview.com/pine-script-reference/v6/#kw_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](https://www.tradingview.com/pine-script-reference/v6/#kw_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](https://www.tradingview.com/pine-script-reference/v6/#kw_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.

The effect of [varip](https://www.tradingview.com/pine-script-reference/v6/#kw_varip) eliminates the [rollback](https://www.tradingview.com/pine-script-docs/language/execution-model/#calculation-based-on-realtime-bars) of variables before each successive execution of a script on the same bar.

**while**

The while statement allows the conditional iteration of a local code block.

SYNTAX

variable\_declaration = while condition

…

continue

…

break

…

return\_expression

where:

**variable\_declaration** - An optional variable declaration. The return expression can provide the initialization value for this variable.

**condition** - when true, the local block of the while statement is executed. When false, execution of the script resumes after the while statement.

**continue** - The continue keyword causes the loop to branch to its next iteration.

**break** - The break keyword causes the loop to terminate. The script's execution resumes after the while statement.

**return\_expression** - An optional line providing the while statement's returning value.

EXAMPLE

//**@version=**6

indicator("while")

// This is a simple example of calculating a factorial using a while loop.

**int** i\_n = input.int(10, "Factorial Size", minval=0)

**int** counter = i\_n

**int** factorial = 1

while counter > 0

factorial := factorial \* counter

counter := counter - 1

plot(factorial)

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.

## **Types**

**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>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new%3Ctype%3E), [array.from](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from) function.

EXAMPLE

//**@version=**6

indicator("array", overlay=true)

**array**<**float**> a = na

a := array.new<**float**>(1, close)

plot(array.get(a, 0))

REMARKS

Array objects are always of "series" form.

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[line](https://www.tradingview.com/pine-script-reference/v6/#type_line)[label](https://www.tradingview.com/pine-script-reference/v6/#type_label)[table](https://www.tradingview.com/pine-script-reference/v6/#type_table)[box](https://www.tradingview.com/pine-script-reference/v6/#type_box)[array.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_array.new%3Ctype%3E)[array.from](https://www.tradingview.com/pine-script-reference/v6/#fun_array.from)

**bool**

Keyword used to explicitly declare the "bool" (boolean) type of a variable or a parameter. "Bool" variables can have values [true](https://www.tradingview.com/pine-script-reference/v6/#const_true) or [false](https://www.tradingview.com/pine-script-reference/v6/#const_false).

EXAMPLE

//**@version=**6

indicator("bool")

**bool** b = true // Same as `b = true`

plot(b ? open : close)

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](https://www.tradingview.com/pine-script-docs/language/type-system/).

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[varip](https://www.tradingview.com/pine-script-reference/v6/#kw_varip)[int](https://www.tradingview.com/pine-script-reference/v6/#type_int)[float](https://www.tradingview.com/pine-script-reference/v6/#type_float)[color](https://www.tradingview.com/pine-script-reference/v6/#type_color)[string](https://www.tradingview.com/pine-script-reference/v6/#type_string)[true](https://www.tradingview.com/pine-script-reference/v6/#const_true)[false](https://www.tradingview.com/pine-script-reference/v6/#const_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](https://www.tradingview.com/pine-script-reference/v6/#fun_box.new) function.

EXAMPLE

//**@version=**6

indicator("box")

// Empty `box1` box ID.

var **box** box1 = na

// `box` type is unnecessary because `box.new()` returns a "box" type.

var box2 = box.new(na, na, na, na)

box3 = box.new(time, open, time + 60 \* 60 \* 24, close, xloc=xloc.bar\_time)

REMARKS

Box objects are always of "series" form.

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[line](https://www.tradingview.com/pine-script-reference/v6/#type_line)[label](https://www.tradingview.com/pine-script-reference/v6/#type_label)[table](https://www.tradingview.com/pine-script-reference/v6/#type_table)[box.new](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_chart.point.from_time), [chart.point.from\_index](https://www.tradingview.com/pine-script-reference/v6/#fun_chart.point.from_index), [chart.point.now](https://www.tradingview.com/pine-script-reference/v6/#fun_chart.point.now), and [chart.point.new](https://www.tradingview.com/pine-script-reference/v6/#fun_chart.point.new) functions.

FIELDS

**index (series int)** The x-coordinate of the point, expressed as a bar index value.

**time (series int)** The x-coordinate of the point, expressed as a UNIX time value, in milliseconds.

**price (series float)** The y-coordinate of the point.

SEE ALSO

[polyline](https://www.tradingview.com/pine-script-reference/v6/#type_polyline)

**color**

Keyword used to explicitly declare the "color" type of a variable or a parameter.

EXAMPLE

//**@version=**6

indicator("color", overlay = true)

**color** textColor = color.green

**color** labelColor = #FF000080 // Red color (FF0000) with 50% transparency (80 which is half of FF).

if barstate.islastconfirmedhistory

label.new(bar\_index, high, text = "Label", color = labelColor, textcolor = textColor)

// When declaring variables with color literals, built-in constants(color.green) or functions (color.new(), color.rgb()), the "color" keyword for the type can be omitted.

c = color.rgb(0,255,0,0)

plot(close, color = c)

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.

Explicitly mentioning the type in a variable declaration is optional, except when it is initialized with [na](https://www.tradingview.com/pine-script-reference/v6/#var_na). Learn more about Pine Script® types in the User Manual page on the [Type System](https://www.tradingview.com/pine-script-docs/language/type-system/).

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[varip](https://www.tradingview.com/pine-script-reference/v6/#kw_varip)[int](https://www.tradingview.com/pine-script-reference/v6/#type_int)[float](https://www.tradingview.com/pine-script-reference/v6/#type_float)[string](https://www.tradingview.com/pine-script-reference/v6/#type_string)[color.rgb](https://www.tradingview.com/pine-script-reference/v6/#fun_color.rgb)[color.new](https://www.tradingview.com/pine-script-reference/v6/#fun_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.

In variable declarations, the compiler can usually infer the qualified type automatically based on the values assigned to a variable, and it can automatically change a variable's qualifier to a stronger one when necessary. The type qualifier hierarchy is "const" < "input" < "simple" < "series", where "const" is the weakest.

Explicitly declaring a variable with the const keyword restricts the type qualifier to "const", meaning the variable cannot accept a value with a stronger qualifier (e.g., "input"), nor can the value assigned to the variable change at any point in the script's execution.

When using this keyword to specify the type qualifier, one must also use a type keyword to declare the allowed type.

SYNTAX

[method ]<functionName>([const <paramType> ]<paramName>[ = <defaultValue>])

[var/varip ]const <variableType> <variableName> = <variableValue>

EXAMPLE

//**@version=**6

indicator("custom plot title")

//**@function** Concatenates two "const string" values.

concatStrings(**const** **string** x, **const** **string** y) =>

**const** **string** result = x + y

//**@variable** The title of the plot.

**const** **string** myTitle = concatStrings("My ", "Plot")

plot(close, myTitle)

EXAMPLE

//**@version=**6

indicator("can't assign input to const")

//**@variable** A variable declared as "const float" that attempts to assign the result of `input.float()` as its value.

// This declaration causes an error. The "input float" qualified type is stronger than "const float".

**const** **float** myVar = input.float(2.0)

plot(myVar)

REMARKS

To learn more, see our User Manual's section on [type qualifiers](https://www.tradingview.com/pine-script-docs/language/type-system/#qualifiers).

SEE ALSO

[simple](https://www.tradingview.com/pine-script-reference/v6/#type_simple)[series](https://www.tradingview.com/pine-script-reference/v6/#type_series)

**float**

Keyword used to explicitly declare the "float" (floating point) type of a variable or a parameter.

EXAMPLE

//**@version=**6

indicator("float")

**float** f = 3.14 // Same as `f = 3.14`

f := na

plot(f)

REMARKS

Explicitly mentioning the type in a variable declaration is optional, except when it is initialized with [na](https://www.tradingview.com/pine-script-reference/v6/#var_na). Learn more about Pine Script® types in the User Manual page on the [Type System](https://www.tradingview.com/pine-script-docs/language/type-system/).

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[varip](https://www.tradingview.com/pine-script-reference/v6/#kw_varip)[int](https://www.tradingview.com/pine-script-reference/v6/#type_int)[bool](https://www.tradingview.com/pine-script-reference/v6/#type_bool)[color](https://www.tradingview.com/pine-script-reference/v6/#type_color)[string](https://www.tradingview.com/pine-script-reference/v6/#type_string)

**int**

Keyword used to explicitly declare the "int" (integer) type of a variable or a parameter.

EXAMPLE

//**@version=**6

indicator("int")

**int** i = 14 // Same as `i = 14`

i := na

plot(i)

REMARKS

Explicitly mentioning the type in a variable declaration is optional, except when it is initialized with [na](https://www.tradingview.com/pine-script-reference/v6/#var_na). Learn more about Pine Script® types in the User Manual page on the [Type System](https://www.tradingview.com/pine-script-docs/language/type-system/).

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[varip](https://www.tradingview.com/pine-script-reference/v6/#kw_varip)[float](https://www.tradingview.com/pine-script-reference/v6/#type_float)[bool](https://www.tradingview.com/pine-script-reference/v6/#type_bool)[color](https://www.tradingview.com/pine-script-reference/v6/#type_color)[string](https://www.tradingview.com/pine-script-reference/v6/#type_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](https://www.tradingview.com/pine-script-reference/v6/#fun_label.new) function.

EXAMPLE

//**@version=**6

indicator("label")

// Empty `label1` label ID.

var **label** label1 = na

// `label` type is unnecessary because `label.new()` returns "label" type.

var label2 = label.new(na, na, na)

if barstate.islastconfirmedhistory

label3 = label.new(bar\_index, high, text = "label3 text")

REMARKS

Label objects are always of "series" form.

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[line](https://www.tradingview.com/pine-script-reference/v6/#type_line)[box](https://www.tradingview.com/pine-script-reference/v6/#type_box)[label.new](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_line.new) function.

EXAMPLE

//**@version=**6

indicator("line")

// Empty `line1` line ID.

var **line** line1 = na

// `line` type is unnecessary because `line.new()` returns "line" type.

var line2 = line.new(na, na, na, na)

line3 = line.new(bar\_index - 1, high, bar\_index, high, extend = extend.right)

REMARKS

Line objects are always of "series" form.

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[label](https://www.tradingview.com/pine-script-reference/v6/#type_label)[box](https://www.tradingview.com/pine-script-reference/v6/#type_box)[line.new](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_linefill.new) function.

EXAMPLE

//**@version=**6

indicator("linefill", overlay=true)

// Empty `linefill1` line ID.

var **linefill** linefill1 = na

// `linefill` type is unnecessary because `linefill.new()` returns "linefill" type.

var linefill2 = linefill.new(na, na, na)

if barstate.islastconfirmedhistory

line1 = line.new(bar\_index - 10, high+1, bar\_index, high+1, extend = extend.right)

line2 = line.new(bar\_index - 10, low+1, bar\_index, low+1, extend = extend.right)

linefill3 = linefill.new(line1, line2, color = color.new(color.green, 80))

REMARKS

Linefill objects are always of "series" form.

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[line](https://www.tradingview.com/pine-script-reference/v6/#type_line)[label](https://www.tradingview.com/pine-script-reference/v6/#type_label)[table](https://www.tradingview.com/pine-script-reference/v6/#type_table)[box](https://www.tradingview.com/pine-script-reference/v6/#type_box)[linefill.new](https://www.tradingview.com/pine-script-reference/v6/#fun_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>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E) function.

EXAMPLE

//**@version=**6

indicator("map", overlay=true)

**map**<**int**, **float**> a = na

a := map.new<**int**, **float**>()

a.put(bar\_index, close)

label.new(bar\_index, a.get(bar\_index), "Current close")

REMARKS

Map objects are always of [series](https://www.tradingview.com/pine-script-docs/language/type-system/#series) form.

SEE ALSO

[map.new<type,type>](https://www.tradingview.com/pine-script-reference/v6/#fun_map.new%3Ctype,type%3E)

**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>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E) function.

EXAMPLE

//**@version=**6

indicator("matrix example")

// Create `m1` matrix of `int` type.

**matrix**<**int**> m1 = matrix.new<**int**>(2, 3, 0)

// `matrix<int>` is unnecessary because the `matrix.new<int>()` function returns an `int` type matrix object.

m2 = matrix.new<**int**>(2, 3, 0)

// Display matrix using a label.

if barstate.islastconfirmedhistory

label.new(bar\_index, high, str.tostring(m2))

REMARKS

Matrix objects are always of "series" form.

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[matrix.new<type>](https://www.tradingview.com/pine-script-reference/v6/#fun_matrix.new%3Ctype%3E)[array](https://www.tradingview.com/pine-script-reference/v6/#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](https://www.tradingview.com/pine-script-reference/v6/#fun_polyline.new) function.

SEE ALSO

[chart.point](https://www.tradingview.com/pine-script-reference/v6/#type_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.

Explicit use of the series keyword when declaring the parameters of a library's exported functions is typically unnecessary, as the compiler can usually automatically detect whether a parameter is compatible with "series" or "simple" qualified values. By default, all exported function parameters are qualified as "series" wherever possible.

In variable declarations, the compiler can usually infer the qualified type automatically based on the values assigned to a variable, and it can automatically change a variable's qualifier to a stronger one when necessary. The type qualifier hierarchy is "const" < "input" < "simple" < "series", where "series" is the strongest.

Explicitly declaring a variable with the series keyword restricts the type qualifier to "series", meaning the script cannot pass its value to any variable or function parameter that requires a value with a weaker qualifier ("const", "input", or "simple").

When using this keyword to specify the type qualifier, one must also use a type keyword to declare the allowed type.

SYNTAX

export [method ]<functionName>([[series ]<paramType>] <paramName>[ = <defaultValue>])

[method ]<functionName>([series <paramType> ]<paramName>[ = <defaultValue>])

[var/varip ]series <variableType> <variableName> = <variableValue>

EXAMPLE

//**@version=**6

//**@description** A library with custom functions.

library("CustomFunctions", overlay = true)

//**@function** Finds the highest `source` value over `length` bars, filtered by the `cond` condition.

export conditionalHighest(**series** **float** source, **series** **bool** cond, **series** **int** length) =>

//**@variable** The highest `source` value from when the `cond` was `true` over `length` bars.

**series** **float** result = na

// Loop to find the highest value.

for i = 0 to length - 1

if cond[i]

value = source[i]

result := math.max(nz(result, value), value)

// Return the `result`.

result

//**@variable** Is `true` once every five bars.

**series** **bool** condition = bar\_index % 5 == 0

//**@variable** The highest `close` value from every fifth bar over the last 100 bars.

**series** **float** hiValue = conditionalHighest(close, condition, 100)

plot(hiValue)

bgcolor(condition ? color.new(color.teal, 80) : na)

EXAMPLE

//**@version=**6

indicator("series variable not allowed")

//**@variable** A variable declared as "series int" with a value of 5.

**series** **int** myVar = 5

// This call causes an error.

// The `histbase` accepts "input int/float". It can't accept the stronger "series int" qualified type.

plot(close, style = plot.style\_histogram, histbase = myVar)

REMARKS

To learn more, see our User Manual's section on [type qualifiers](https://www.tradingview.com/pine-script-docs/language/type-system/#qualifiers).

SEE ALSO

[simple](https://www.tradingview.com/pine-script-reference/v6/#type_simple)[const](https://www.tradingview.com/pine-script-reference/v6/#type_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.

To restrict the parameters in a library's exported functions to only allow values with a "simple" or weaker type qualifier, using the simple keyword when declaring parameters is often necessary, as libraries automatically qualify all parameters as "series" wherever possible by default. Explicitly restricting functions to accept "simple" arguments also allows them to return "simple" values in some cases, depending on the operations they execute, making them usable with the parameters of built-in functions that do not allow "series" arguments.

In variable declarations, the compiler can usually infer the qualified type automatically based on the values assigned to a variable, and it can automatically change a variable's qualifier to a stronger one when necessary. The type qualifier hierarchy is "const" < "input" < "simple" < "series", where "simple" is stronger than "input" and "const".

Explicitly declaring a variable with the simple keyword restricts the type qualifier to "simple", meaning the script cannot pass its value to any variable or function parameter that requires a value with a weaker qualifier ("const" or "input"). Additionally, one cannot assign a "series" value to a variable explicitly declared with the simple keyword.

When using this keyword to specify the type qualifier, one must also use a type keyword to declare the allowed type.

SYNTAX

export [method ]<functionName>([[simple ]<paramType>] <paramName>[ = <defaultValue>])

[method ]<functionName>([simple <paramType> ]<paramName>[ = <defaultValue>])

[var/varip ]simple <variableType> <variableName> = <variableValue></variableValue>

EXAMPLE

//**@version=**6

//**@description** A library with custom functions.

library("CustomFunctions", overlay = true)

//**@function** Calculates the length values for a ribbon of four EMAs by multiplying the `baseLength`.

//**@param** baseLength The initial EMA length. Requires "simple int" because you can't use "series int" in `ta.ema()`.

//**@returns** A tuple of length values.

export ribbonLengths(**simple** **int** baseLength) =>

**simple** **int** length1 = baseLength

**simple** **int** length2 = baseLength \* 2

**simple** **int** length3 = baseLength \* 3

**simple** **int** length4 = baseLength \* 4

[length1, length2, length3, length4]

// Get a tuple of "simple int" length values.

[len1, len2, len3, len4] = ribbonLengths(14)

// Plot four EMAs using the values from the tuple.

plot(ta.ema(close, len1), "EMA 1", color = color.red)

plot(ta.ema(close, len2), "EMA 1", color = color.orange)

plot(ta.ema(close, len3), "EMA 1", color = color.green)

plot(ta.ema(close, len4), "EMA 1", color = color.blue)

EXAMPLE

//**@version=**6

indicator("can't change simple to series")

//**@variable** A variable declared as "simple float" with a value of 5.0.

**simple** **float** myVar = 5.0

// This reassignment causes an error.

// The `close` variable returns a "series float" value. Since `myVar` is restricted to "simple" values, it cannot

// change its qualifier to "series".

myVar := close

plot(myVar)

REMARKS

To learn more, see our User Manual's section on [type qualifiers](https://www.tradingview.com/pine-script-docs/language/type-system/#qualifiers).

SEE ALSO

[series](https://www.tradingview.com/pine-script-reference/v6/#type_series)[const](https://www.tradingview.com/pine-script-reference/v6/#type_const)

**string**

Keyword used to explicitly declare the "string" type of a variable or a parameter.

EXAMPLE

//**@version=**6

indicator("string")

**string** s = "Hello World!" // Same as `s = "Hello world!"`

// string s = na // same as ""

plot(na, title=s)

REMARKS

Explicitly mentioning the type in a variable declaration is optional, except when it is initialized with [na](https://www.tradingview.com/pine-script-reference/v6/#var_na). Learn more about Pine Script® types in the User Manual page on the [Type System](https://www.tradingview.com/pine-script-docs/language/type-system/).

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[varip](https://www.tradingview.com/pine-script-reference/v6/#kw_varip)[int](https://www.tradingview.com/pine-script-reference/v6/#type_int)[float](https://www.tradingview.com/pine-script-reference/v6/#type_float)[bool](https://www.tradingview.com/pine-script-reference/v6/#type_bool)[str.tostring](https://www.tradingview.com/pine-script-reference/v6/#fun_str.tostring)[str.format](https://www.tradingview.com/pine-script-reference/v6/#fun_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](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new) function.

EXAMPLE

//**@version=**6

indicator("table")

// Empty `table1` table ID.

var **table** table1 = na

// `table` type is unnecessary because `table.new()` returns "table" type.

var table2 = table.new(position.top\_left, na, na)

if barstate.islastconfirmedhistory

var table3 = table.new(position = position.top\_right, columns = 1, rows = 1, bgcolor = color.yellow, border\_width = 1)

table.cell(table\_id = table3, column = 0, row = 0, text = "table3 text")

REMARKS

Table objects are always of "series" form.

SEE ALSO

[var](https://www.tradingview.com/pine-script-reference/v6/#kw_var)[line](https://www.tradingview.com/pine-script-reference/v6/#type_line)[label](https://www.tradingview.com/pine-script-reference/v6/#type_label)[box](https://www.tradingview.com/pine-script-reference/v6/#type_box)[table.new](https://www.tradingview.com/pine-script-reference/v6/#fun_table.new)

## **Operators**

**-**

Subtraction or unary minus. Applicable to numerical expressions.

SYNTAX

expr1 - expr2

RETURNS

Returns integer or float value, or series of values:

Binary - returns expr1 minus expr2.

Unary - returns the negation of expr.

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

expr1 -= expr2

EXAMPLE

//**@version=**6

indicator("-=")

// Equals to expr1 = expr1 - expr2.

a = 2

b = 3

a -= b

// Result: a = -1.

plot(a)

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

<var\_name> := <new\_value>

EXAMPLE

//**@version=**6

indicator("My script")

myVar = 10

if close > open

// Modifies the existing global scope `myVar` variable by changing its value from 10 to 20.

myVar := 20

// Creates a new `myVar` variable local to the `if` condition and unreachable from the global scope.

// Does not affect the `myVar` declared in global scope.

myVar = 30

plot(myVar)

**!=**

Not equal to. Applicable to expressions of any type.

SYNTAX

expr1 != expr2

RETURNS

Boolean value, or series of boolean values.

**?:**

Ternary conditional operator.

SYNTAX

expr1 ? expr2 : expr3

EXAMPLE

//**@version=**6

indicator("?:")

// Draw circles at the bars where open crosses close

s2 = ta.cross(open, close) ? math.avg(open,close) : na

plot(s2, style=plot.style\_circles, linewidth=2, color=color.red)

// Combination of ?: operators for 'switch'-like logic

c = timeframe.isintraday ? color.red : timeframe.isdaily ? color.green : timeframe.isweekly ? color.blue : color.gray

plot(hl2, color=c)

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](https://www.tradingview.com/pine-script-reference/v6/#var_na) for 'else' branch if you do not need it.

You can combine two or more [?:](https://www.tradingview.com/pine-script-reference/v6/#op_?:) operators to achieve the equivalent of a 'switch'-like statement (see examples above).

You may use arithmetic operators with numbers as well as with series variables. In case of usage with series the operators are applied elementwise.

SEE ALSO

[na](https://www.tradingview.com/pine-script-reference/v6/#var_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

expr1[expr2]

EXAMPLE

//**@version=**6

indicator("[]")

// [] can be used to "save" variable value between bars

a = 0.0 // declare `a`

a := a[1] // immediately set current value to the same as previous. `na` in the beginning of history

if high == low // if some condition - change `a` value to another

a := low

plot(a)

RETURNS

A series of values.

SEE ALSO

[math.floor](https://www.tradingview.com/pine-script-reference/v6/#fun_math.floor)

**\***

Multiplication. Applicable to numerical expressions.

SYNTAX

expr1 \* expr2

RETURNS

Integer or float value, or series of values.

**\*=**

Multiplication assignment. Applicable to numerical expressions.

SYNTAX

expr1 \*= expr2

EXAMPLE

//**@version=**6

indicator("\*=")

// Equals to expr1 = expr1 \* expr2.

a = 2

b = 3

a \*= b

// Result: a = 6.

plot(a)

RETURNS

Integer or float value, or series of values.

**/**

Division. Applicable to numerical expressions.

SYNTAX

expr1 / expr2

RETURNS

Integer or float value, or series of values.

**/=**

Division assignment. Applicable to numerical expressions.

SYNTAX

expr1 /= expr2

EXAMPLE

//**@version=**6

indicator("/=")

// Equals to expr1 = expr1 / expr2.

**float** a = 3.0

b = 3

a /= b

// Result: a = 1.

plot(a)

RETURNS

Integer or float value, or series of values.

**%**

Modulo (integer remainder). Applicable to numerical expressions.

SYNTAX

expr1 % expr2

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.

Example: -1 % 9 = -1 - 9 \* int(-1/9) = -1 - 9 \* int(-0.111) = -1 - 9 \* 0 = -1.

**%=**

Modulo assignment. Applicable to numerical expressions.

SYNTAX

expr1 %= expr2

EXAMPLE

//**@version=**6

indicator("%=")

// Equals to expr1 = expr1 % expr2.

a = 3

b = 3

a %= b

// Result: a = 0.

plot(a)

RETURNS

Integer or float value, or series of values.

**+**

Addition or unary plus. Applicable to numerical expressions or strings.

SYNTAX

expr1 + expr2

RETURNS

Binary + for strings returns concatenation of expr1 and expr2

For numbers returns integer or float value, or series of values:

Binary + returns expr1 plus expr2.

Unary + returns expr (does nothing added just for the symmetry with the unary - operator).

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

expr1 += expr2

EXAMPLE

//**@version=**6

indicator("+=")

// Equals to expr1 = expr1 + expr2.

a = 2

b = 3

a += b

// Result: a = 5.

plot(a)

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

expr1 < expr2

RETURNS

Boolean value, or series of boolean values.

**<=**

Less than or equal to. Applicable to numerical expressions.

SYNTAX

expr1 <= expr2

RETURNS

Boolean value, or series of boolean values.

**==**

Equal to. Applicable to expressions of any type.

SYNTAX

expr1 == expr2

RETURNS

Boolean value, or series of boolean values.

**=>**

The '=>' operator is used in user-defined function declarations and in [switch](https://www.tradingview.com/pine-script-reference/v6/#kw_switch) statements.

The function declaration syntax is:

SYNTAX

<identifier>([<parameter\_name>[=<default\_value>]], ...) =>

<local\_block>

<function\_result>

A <local\_block> is zero or more Pine Script® statements.

The <function\_result> is a variable, an expression, or a tuple.

EXAMPLE

//**@version=**6

indicator("=>")

// single-line function

f1(x, y) => x + y

// multi-line function

f2(x, y) =>

sum = x + y

sumChange = ta.change(sum, 10)

// Function automatically returns the last expression used in it

plot(f1(30, 8) + f2(1, 3))

REMARKS

You can learn more about user-defined functions in the User Manual's pages on [Declaring functions](https://www.tradingview.com/pine-script-docs/language/user-defined-functions/) and [Libraries](https://www.tradingview.com/pine-script-docs/concepts/libraries/).

**>**

Greater than. Applicable to numerical expressions.

SYNTAX

expr1 > expr2

RETURNS

Boolean value, or series of boolean values.

**>=**

Greater than or equal to. Applicable to numerical expressions.

SYNTAX

expr1 >= expr2

RETURNS

Boolean value, or series of boolean values.

## **Annotations**

**@description**

Sets a custom description for scripts that use the [library](https://www.tradingview.com/pine-script-reference/v6/#fun_library) declaration statement. The text provided with this annotation will be used to pre-fill the "Description" field in the publication dialogue.

EXAMPLE

//**@version=**6

// **@description** Provides a tool to quickly output a label on the chart.

library("MyLibrary")

// **@function** Outputs a label with `labelText` on the bar's high.

// **@param** labelText (series string) The text to display on the label.

// **@returns** Drawn label.

export drawLabel(**string** labelText) =>

label.new(bar\_index, high, text = labelText)

**@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](https://www.tradingview.com/pine-script-reference/v6/#kw_export) keyword will pre-fill the "Description" field in the publication dialogue.

EXAMPLE

//**@version=**6

indicator("Session highlight", overlay = true)

//**@enum** Contains fields with popular timezones as titles.

//**@field** exch Has an empty string as the title to represent the chart timezone.

enum tz

utc = "UTC"

exch = ""

ny = "America/New\_York"

chi = "America/Chicago"

lon = "Europe/London"

tok = "Asia/Tokyo"

//**@variable** The session string.

selectedSession = input.session("1200-1500", "Session")

//**@variable** The selected timezone. The input's dropdown contains the fields in the `tz` enum.

selectedTimezone = input.enum(tz.utc, "Session Timezone")

//**@variable** Is `true` if the current bar's time is in the specified session.

**bool** inSession = false

if not na(time("", selectedSession, str.tostring(selectedTimezone)))

inSession := true

// Highlight the background when `inSession` is `true`.

bgcolor(inSession ? color.new(color.green, 90) : na, title = "Active session highlight")

**@field**

If placed above a [type](https://www.tradingview.com/pine-script-reference/v6/#kw_type) or [enum](https://www.tradingview.com/pine-script-reference/v6/#kw_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.

The Pine Editor's autosuggest uses this description and displays it when a user hovers over the type/enum or field name. When used in [library](https://www.tradingview.com/pine-script-reference/v6/#fun_library) scripts, the descriptions of all types/enums using the [export](https://www.tradingview.com/pine-script-reference/v6/#kw_export) keyword will pre-fill the "Description" field in the publication dialogue.

EXAMPLE

//**@version=**6

indicator("New high over the last 20 bars", overlay = true)

//**@type** A point on a chart.

//**@field** index The index of the bar where the point is located, i.e., its `x` coordinate.

//**@field** price The price where the point is located, i.e., its `y` coordinate.

type Point

**int** index

**float** price

//**@variable** If the current `high` is the highest over the last 20 bars, returns a new `Point` instance, `na` otherwise.

**Point** highest = na

if ta.highestbars(high, 20) == 0

highest := Point.new(bar\_index, high)

label.new(highest.index, highest.price, str.tostring(highest.price))

**@function**

If placed above a function declaration, it adds a custom description for the function.

The Pine Editor's autosuggest uses this description and displays it when a user hovers over the function name. When used in [library](https://www.tradingview.com/pine-script-reference/v6/#fun_library) scripts, the descriptions of all functions using the [export](https://www.tradingview.com/pine-script-reference/v6/#kw_export) keyword will pre-fill the "Description" field in the publication dialogue.

EXAMPLE

//**@version=**6

// **@description** Provides a tool to quickly output a label on the chart.

library("MyLibrary")

// **@function** Outputs a label with `labelText` on the bar's high.

// **@param** labelText (series string) The text to display on the label.

// **@returns** Drawn label.

export drawLabel(**string** labelText) =>

label.new(bar\_index, high, text = labelText)

**@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.

The Pine Editor's autosuggest uses this description and displays it when a user hovers over the function name. When used in [library](https://www.tradingview.com/pine-script-reference/v6/#fun_library) scripts, the descriptions of all functions using the [export](https://www.tradingview.com/pine-script-reference/v6/#kw_export) keyword will pre-fill the "Description" field in the publication dialogue.

EXAMPLE

//**@version=**6

// **@description** Provides a tool to quickly output a label on the chart.

library("MyLibrary")

// **@function** Outputs a label with `labelText` on the bar's high.

// **@param** labelText (series string) The text to display on the label.

// **@returns** Drawn label.

export drawLabel(**string** labelText) =>

label.new(bar\_index, high, text = labelText)

**@returns**

If placed above a function declaration, it adds a custom description for what that function returns.

The Pine Editor's autosuggest uses this description and displays it when a user hovers over the function name. When used in [library](https://www.tradingview.com/pine-script-reference/v6/#fun_library) scripts, the descriptions of all functions using the [export](https://www.tradingview.com/pine-script-reference/v6/#kw_export) keyword will pre-fill the "Description" field in the publication dialogue.

EXAMPLE

//**@version=**6

// **@description** Provides a tool to quickly output a label on the chart.

library("MyLibrary")

// **@function** Outputs a label with `labelText` on the bar's high.

// **@param** labelText (series string) The text to display on the label.

// **@returns** Drawn label.

export drawLabel(**string** labelText) =>

label.new(bar\_index, high, text = labelText)

**@strategy\_alert\_message**

If used within a [strategy](https://www.tradingview.com/pine-script-reference/v6/#fun_strategy) script, it provides a default message to pre-fill the "Message" field in the alert creation dialogue.

EXAMPLE

//**@version=**6

strategy("My strategy", overlay=true, margin\_long=100, margin\_short=100)

//**@strategy**\_alert\_message Strategy alert on symbol {{ticker}}

longCondition = ta.crossover(ta.sma(close, 14), ta.sma(close, 28))

if (longCondition)

strategy.entry("My Long Entry Id", strategy.long)

strategy.exit("Exit", "My Long Entry Id", profit = 10 / syminfo.mintick, loss = 10 / syminfo.mintick)

**@type**

If placed above a type declaration, it adds a custom description for the type.

The Pine Editor's autosuggest uses this description and displays it when a user hovers over the type name. When used in [library](https://www.tradingview.com/pine-script-reference/v6/#fun_library) scripts, the descriptions of all types using the [export](https://www.tradingview.com/pine-script-reference/v6/#kw_export) keyword will pre-fill the "Description" field in the publication dialogue.

EXAMPLE

//**@version=**6

indicator("New high over the last 20 bars", overlay = true)

//**@type** A point on a chart.

//**@field** index The index of the bar where the point is located, i.e., its `x` coordinate.

//**@field** price The price where the point is located, i.e., its `y` coordinate.

type Point

**int** index

**float** price

//**@variable** If the current `high` is the highest over the last 20 bars, returns a new `Point` instance, `na` otherwise.

**Point** highest = na

if ta.highestbars(high, 20) == 0

highest := Point.new(bar\_index, high)

label.new(highest.index, highest.price, str.tostring(highest.price))

**@variable**

If placed above a variable declaration, it adds a custom description for the variable.

The Pine Editor's autosuggest uses this description and displays it when a user hovers over the variable name.

EXAMPLE

//**@version=**6

indicator("New high over the last 20 bars", overlay = true)

//**@type** A point on a chart.

//**@field** index The index of the bar where the point is located, i.e., its `x` coordinate.

//**@field** price The price where the point is located, i.e., its `y` coordinate.

type Point

**int** index

**float** price

//**@variable** If the current `high` is the highest over the last 20 bars, returns a new `Point` instance, `na` otherwise.

**Point** highest = na

if ta.highestbars(high, 20) == 0

highest := Point.new(bar\_index, high)

label.new(highest.index, highest.price, str.tostring(highest.price))

**@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

//**@version=**6

indicator("Pine v6 Indicator")

plot(close)

EXAMPLE

//This indicator has no version annotation, so it will try to use v1.

//Pine Script® v1 has no function named `indicator()`, so the script will not compile.

indicator("Pine v1 Indicator")

plot(close)

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.