Common Functions & Parameters

Annotation Functions:

`indicator(title="Example", shorttitle="EMA", overlay=true)`	Defines your script as a standard 'indicator' script
`strategy(title="Strategy", initial_capital=1000, currency="USD", calc_on_order_fills=true, calc_on_every_tick=true)`	Defines your script as a strategy script, giving you access to strategy variables & functions
`library(title="MyLibrary", overlay=true)`	Defines your script as a library with exportable functions

User Input Functions:

`input.int(title="Number", defval=5, minval=0, maxval=10, step=2)`	Used to get whole number inputs from the user
`input.float(title="Number", defval=5.0, minval=0.0, maxval=10.0, step=2.5)`	Used to get decimal number inputs from the user
`input.bool(title="Yes/No", defval=true)`	Used to get true/false Boolean inputs from the user
`input.source(title="Symbol", defval="OANDA:EURUSD")`	Used to get market ticker code inputs from the user
`input.price(title= "Price", defval=1.05, confirm=true)`	Used to get a price (float) input from the user

Drawing Functions:

`plot(close, title="Price Value", style=plot.style_linebr)`	Used to plot numbers (prices, indicator values etc) to chart
`bgcolor(color.new(color.red,50))`	Used to change the background color of your chart
`plotshape(close > open, style=shape.triangleup, color=color.green)`	Used to plot shapes onto the chart (type shape.CTRL_SPACE in Pine editor to see full list of possible shapes)
`hline(70.0)`	Used to draw a horizontal line across your chart/indicator

Common Functions & Parameters

Technical Analysis Functions:

ta.atr(14)	Gets the current ATR value with a 14-period lookback
ta.ema(close, 50) / ta.sma(close, 50)	Gets a 50-period Moving Average based on candle closes
ta.rsi(close, 14)	Gets a 14-period RSI value based on candle closes
ta.vwap	Gets the volume-weighted average price for the current bar
ta.highest(high, 7)	Gets the highest high over the past 7 bars
ta.lowest(low, 10)	Gets the lowest low over the past 10 bars
ta.barssince(close > open)	Counts how many bars printed since the condition was true
ta.cross(value1, value2)	Returns true if value1 crosses over/under value2
ta.bb(close, 20, 2)	Returns a tuple containing Bollinger Band values eg. [mid, upper, lower] = ta.bb(source, length, stdeviation)
ta.macd(close, 12, 26, 9)	Returns a tuple contains MACD values eg. [macd, signal, hist] = ta.macd(src, fast, slow, siglen)

Bar States:

barstate.isconfirmed	True on historical bars and on final tick of real-time bar
barstate.isfirst	True on very first bar on the chart
barstate.ishistory	True on historical bars, false on real-time bar
barstate.isnew	True on historical bars and on first tick of a new bar
barstate.islastconfirmedhistory	True on the most recent confirmed historical bar
barstate.islast	True on the final bar on chart (will not return true on strategy scripts unless recalculate on every tick is on)
barstate.isrealtime	True on the current real-time bar, false on historical bars, false on strategy scripts unless recalc_on_every_tick is on

Common Functions & Parameters

Symbol Info:

syminfo.basecurrency	Returns the base currency (eg. BTC/USD returns BTC)
syminfo.currency	Returns the quote currency (eg. EUR/USD returns EUR)
syminfo.description	Returns symbol description (Eg. CL1! Returns contract name)
syminfo.mintick	Returns the minimum tick movement for the symbol (eg. 0.001)
syminfo.pointvalue	Returns the value of a point (eg. CL1! is 1000, FX is 1)
syminfo.prefix	Returns symbol prefix (ie. exchange, BINANCE:, OANDA: etc)
syminfo.root	Returns root contract for futures (eg. CLM2014 is CL)
syminfo.session	Returns session type (regular or extended hours for stocks)
syminfo.ticker	Returns symbol name without exchange prefix (eg. AAPL)
syminfo.tickerid	Returns symbol name WITH exchange (eg. NASDAQ:AAPL)
syminfo.timezone	Returns the time-zone of the exchange as a ‘timestamp’
syminfo.type	Returns market type (stock,futures,index,forex,crypto etc)

String Functions:

str.tostring(123.5)	Converts the given number into text (string data type)
str.tostring(close, “#.#####”)	Formats the string (eg. using “#.#” will round to 1 decimal)
str.format(“{0,number,currency)”, 1.34)	Formats the given number to look like $1.34
str.contains(“NASDAQ:AAPL”, “NASDAQ”)	Returns true if the second string is found in the first

Common Functions & Parameters

Request Functions:

`request.security(syminfo.tickerid, “D”, high[barstate.isconfirmed ? 0 : 1])`	Returns the current market’s daily high with repainting behavior eliminated
`request.security(“NASDAQ:AAPL”, “D”, close)`	Returns Apple stock’s current close (will repaint)
`request.quandl(“FRED/DFF”, barmerge.gaps_off, 0)`	Returns current Effect Federal Funds Rate
`request.financial(syminfo.tickerid, “ACCOUNTS_PAYABLE”, “FQ”, gaps=barmerge.gaps_off)`	Returns the accounts payable amount for the current symbol and the current financial quarter

Strategy Functions:

`strategy.entry(id=“Long”, direction=strategy.long, qty=1000, comment= “Yay”)`	Enters a trade with the ID of Long, the direction of Long & the position size of 1000 contracts/units and text of Yay
`strategy.exit(id=“Long Exit”, from_entry=“Long”, limit=targetPrice, stop=stopPrice, when=strategy.position_size > 0)`	Exits the trade with id “Long” if price exceeds the take-profit limit or stop loss price, and only executes if there are open long contracts (pos_size will be < 0 for shorts)
`strategy.close_all(when=strategy.position_size != 0, alert_message= “Closed trade at {{close}}”)`	Closes all trades only when a trade is open, and includes an alert message that includes the closing price
`strategy.close(id= “Exit”, when=strategy.position_size < 0)`	Closes any open short trades
`strategy.position_size`	Returns open contracts (0 = flat, > 0 = long, < 0 = short)
`strategy.equity`	Returns current account balance including historical P&L
`strategy.initial_capital`	Returns starting account balance
`strategy.closedtrades`	Returns how many trades have been taken so far during test
`strategy.wintrades`	Returns how many trades have won so far during testing
`strategy.losstrades`	Returns how many trades have lost so far during testing

Common Functions & Parameters

Math Functions:

math.abs(-100)	- returns 100	Returns any given number as a positive number
math.avg(10, 23, 15, 33)	- returns 20.25	Returns the average over the given number set
math.round(23.5)	- returns 24	Returns the given number rounded to the nearest whole number
math.round(100.2346, 3)	- returns 100.235	Returns the number rounded to the given decimal precision
math.floor(10.3)	- returns 10	Returns largest integer less than or equal the given number
math.max(10, 50, 3, 2)	- returns 50	Returns maximum number in given number set
math.min(10, 50, 3, 2)	- returns 2	Returns minimum number in given number set
math.random(0, 100)		Returns a random number between the two given inputs

Drawing Object Functions:

lb = label.new(bar_index, high, “Text”,)	Creates a new label object attached to bar high
label.delete(lb[1])	Deletes the given label from the given bar offset
tb = table.new(position_top_right, 1, 1, color.black)	Creates a new table in top-right with 1 row & 1 column
table.cell(tb, 0, 0, “Close: ” + str.tostring(close))	Fills given row & column cell with given text on given table
line = line.new(bar_index - 10, low[10], bar_index, high)	Draws a line from the low 10 bars ago to current bar high

Miscellaneous:

import ZenAndTheArtOfTrading/ZenLibrary/3 as zen	Imports ZenLibrary v3 as ‘zen’ so we can use zen.* functions
var <persistent_variable>	Creates a variable which saves its value over all bars
varip <persistent_intrabar_variable>	Creates a variable which saves its value over realtime ticks
var float t_stop = na	Creates a float called ‘t_stop’ with an initial value of na
int[] array = array.from(1, 2, 3)	Creates an integer array from the given values