5 Module talib.func

5.1 Functions

___bootstrap___()

ACOS(*real*)

```
Vector Trigonometric ACos (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

AD(*high*, *low*, *close*, *volume*)

```
Chaikin A/D Line (Volume Indicators)

Inputs:
    prices: ['high', 'low', 'close', 'volume']
Outputs:
    real
```

ADD(*real0*, *real1*)

```
Vector Arithmetic Add (Math Operators)

Inputs:
    real0: (any ndarray)
    real1: (any ndarray)
Outputs:
    real
```

ADOSC(*high*, *low*, *close*, *volume*, *fastperiod*=?, *slowperiod*=?)

```
Chaikin A/D Oscillator (Volume Indicators)

Inputs:
    prices: ['high', 'low', 'close', 'volume']
Parameters:
    fastperiod: 3
    slowperiod: 10
Outputs:
    real
```

ADX(*high*, *low*, *close*, *timeperiod*=?)

```
Average Directional Movement Index (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    timeperiod: 14
Outputs:
    real
```

ADXR(*high*, *low*, *close*, *timeperiod*=?)

```
Average Directional Movement Index Rating (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    timeperiod: 14
Outputs:
    real
```

APO(*real*, *fastperiod*=?, *slowperiod*=?, *matype*=?)

```
Absolute Price Oscillator (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    fastperiod: 12
    slowperiod: 26
    matype: 0 (Simple Moving Average)
Outputs:
    real
```

AROON(*high*, *low*, *timeperiod*=?)

```
Aroon (Momentum Indicators)

Inputs:
    prices: ['high', 'low']
Parameters:
    timeperiod: 14
Outputs:
    aroondown
    aroonup
```

AROONOSC(*high*, *low*, *timeperiod*=?)

```
Aroon Oscillator (Momentum Indicators)

Inputs:
    prices: ['high', 'low']
Parameters:
    timeperiod: 14
Outputs:
    real
```

ASIN(*real*)

```
Vector Trigonometric ASin (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

ATAN(*real*)

```
Vector Trigonometric ATan (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

ATR(*high*, *low*, *close*, *timeperiod*=?)

```
Average True Range (Volatility Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    timeperiod: 14
Outputs:
    real
```

AVGPRICE(*open*, *high*, *low*, *close*)

```
Average Price (Price Transform)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    real
```

BBANDS(*real*, *timeperiod*=?, *nbdevup*=?, *nbdevdn*=?, *matype*=?)

```
Bollinger Bands (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 5
    nbdevup: 2
    nbdevdn: 2
    matype: 0 (Simple Moving Average)
Outputs:
    upperband
    middleband
    lowerband
```

BETA(*real0*, *real1*, *timeperiod*=?)

```
Beta (Statistic Functions)

Inputs:
    real0: (any ndarray)
    real1: (any ndarray)
Parameters:
    timeperiod: 5
Outputs:
    real
```

BOP(*open*, *high*, *low*, *close*)

```
Balance Of Power (Momentum Indicators)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    real
```

CCI(*high*, *low*, *close*, *timeperiod*=?)

```
Commodity Channel Index (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    timeperiod: 14
Outputs:
    real
```

CDL2CROWS(*open*, *high*, *low*, *close*)

```
Two Crows (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDL3BLACKCROWS(*open*, *high*, *low*, *close*)

```
Three Black Crows (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDL3INSIDE(*open*, *high*, *low*, *close*)

```
Three Inside Up/Down (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDL3LINESTRIKE(*open*, *high*, *low*, *close*)

```
Three-Line Strike  (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDL3OUTSIDE(*open*, *high*, *low*, *close*)

```
Three Outside Up/Down (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDL3STARSINSOUTH(*open*, *high*, *low*, *close*)

```
Three Stars In The South (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDL3WHITESOLDIERS(*open*, *high*, *low*, *close*)

```
Three Advancing White Soldiers (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLABANDONEDBABY(*open*, *high*, *low*, *close*, *penetration*=?)

```
Abandoned Baby (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Parameters:
    penetration: 0.3
Outputs:
    integer (values are -100, 0 or 100)
```

CDLADVANCEBLOCK(*open*, *high*, *low*, *close*)

```
Advance Block (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLBELTHOLD(*open*, *high*, *low*, *close*)

```
Belt-hold (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLBREAKAWAY(*open*, *high*, *low*, *close*)

```
Breakaway (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLCLOSINGMARUBOZU(*open*, *high*, *low*, *close*)

```
Closing Marubozu (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLCONCEALBABYSWALL(*open*, *high*, *low*, *close*)

```
Concealing Baby Swallow (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLCOUNTERATTACK(*open*, *high*, *low*, *close*)

```
Counterattack (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLDARKCLOUDCOVER(*open*, *high*, *low*, *close*, *penetration*=?)

```
Dark Cloud Cover (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Parameters:
    penetration: 0.5
Outputs:
    integer (values are -100, 0 or 100)
```

CDLDOJI(*open*, *high*, *low*, *close*)

```
Doji (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLDOJISTAR(*open*, *high*, *low*, *close*)

```
Doji Star (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLDRAGONFLYDOJI(*open*, *high*, *low*, *close*)

```
Dragonfly Doji (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLENGULFING(*open*, *high*, *low*, *close*)

```
Engulfing Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLEVENINGDOJISTAR(*open*, *high*, *low*, *close*, *penetration*=?)

```
Evening Doji Star (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Parameters:
    penetration: 0.3
Outputs:
    integer (values are -100, 0 or 100)
```

CDLEVENINGSTAR(*open*, *high*, *low*, *close*, *penetration*=?)

```
Evening Star (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Parameters:
    penetration: 0.3
Outputs:
    integer (values are -100, 0 or 100)
```

CDLGAPSIDESIDEWHITE(*open*, *high*, *low*, *close*)

```
Up/Down-gap side-by-side white lines (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLGRAVESTONEDOJI(*open*, *high*, *low*, *close*)

```
Gravestone Doji (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLHAMMER(*open*, *high*, *low*, *close*)

```
Hammer (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLHANGINGMAN(*open*, *high*, *low*, *close*)

```
Hanging Man (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLHARAMI(*open*, *high*, *low*, *close*)

```
Harami Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLHARAMICROSS(*open*, *high*, *low*, *close*)

```
Harami Cross Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLHIGHWAVE(*open*, *high*, *low*, *close*)

```
High-Wave Candle (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLHIKKAKE(*open*, *high*, *low*, *close*)

```
Hikkake Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLHIKKAKEMOD(*open*, *high*, *low*, *close*)

```
Modified Hikkake Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLHOMINGPIGEON(*open*, *high*, *low*, *close*)

```
Homing Pigeon (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLIDENTICAL3CROWS(*open*, *high*, *low*, *close*)

```
Identical Three Crows (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLINNECK(*open*, *high*, *low*, *close*)

```
In-Neck Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLINVERTEDHAMMER(*open*, *high*, *low*, *close*)

```
Inverted Hammer (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLKICKING(*open*, *high*, *low*, *close*)

```
Kicking (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLKICKINGBYLENGTH(*open*, *high*, *low*, *close*)

```
Kicking - bull/bear determined by the longer marubozu (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLLADDERBOTTOM(*open*, *high*, *low*, *close*)

```
Ladder Bottom (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLLONGLEGGEDDOJI(*open*, *high*, *low*, *close*)

```
Long Legged Doji (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLLONGLINE(*open*, *high*, *low*, *close*)

```
Long Line Candle (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLMARUBOZU(*open*, *high*, *low*, *close*)

```
Marubozu (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLMATCHINGLOW(*open*, *high*, *low*, *close*)

```
Matching Low (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLMATHOLD(*open*, *high*, *low*, *close*, *penetration*=?)

```
Mat Hold (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Parameters:
    penetration: 0.5
Outputs:
    integer (values are -100, 0 or 100)
```

CDLMORNINGDOJISTAR(*open*, *high*, *low*, *close*, *penetration*=?)

```
Morning Doji Star (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Parameters:
    penetration: 0.3
Outputs:
    integer (values are -100, 0 or 100)
```

CDLMORNINGSTAR(*open*, *high*, *low*, *close*, *penetration*=?)

```
Morning Star (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Parameters:
    penetration: 0.3
Outputs:
    integer (values are -100, 0 or 100)
```

CDLONNECK(*open*, *high*, *low*, *close*)

```
On-Neck Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLPIERCING(*open*, *high*, *low*, *close*)

```
Piercing Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLRICKSHAWMAN(*open*, *high*, *low*, *close*)

```
Rickshaw Man (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLRISEFALL3METHODS(*open*, *high*, *low*, *close*)

```
Rising/Falling Three Methods (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLSEPARATINGLINES(*open*, *high*, *low*, *close*)

```
Separating Lines (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLSHOOTINGSTAR(*open*, *high*, *low*, *close*)

```
Shooting Star (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLSHORTLINE(*open*, *high*, *low*, *close*)

```
Short Line Candle (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLSPINNINGTOP(*open*, *high*, *low*, *close*)

```
Spinning Top (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLSTALLEDPATTERN(*open*, *high*, *low*, *close*)

```
Stalled Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLSTICKSANDWICH(*open*, *high*, *low*, *close*)

```
Stick Sandwich (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLTAKURI(*open*, *high*, *low*, *close*)

```
Takuri (Dragonfly Doji with very long lower shadow) (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLTASUKIGAP(*open*, *high*, *low*, *close*)

```
Tasuki Gap (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLTHRUSTING(*open*, *high*, *low*, *close*)

```
Thrusting Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLTRISTAR(*open*, *high*, *low*, *close*)

```
Tristar Pattern (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLUNIQUE3RIVER(*open*, *high*, *low*, *close*)

```
Unique 3 River (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLUPSIDEGAP2CROWS(*open*, *high*, *low*, *close*)

```
Upside Gap Two Crows (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CDLXSIDEGAP3METHODS(*open*, *high*, *low*, *close*)

```
Upside/Downside Gap Three Methods (Pattern Recognition)

Inputs:
    prices: ['open', 'high', 'low', 'close']
Outputs:
    integer (values are -100, 0 or 100)
```

CEIL(*real*)

```
Vector Ceil (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

CMO(*real*, *timeperiod*=?)

```
Chande Momentum Oscillator (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 14
Outputs:
    real
```

CORREL(*real0*, *real1*, *timeperiod*=?)

```
Pearson's Correlation Coefficient (r) (Statistic Functions)

Inputs:
    real0: (any ndarray)
    real1: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

COS(*real*)

```
Vector Trigonometric Cos (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

COSH(*real*)

```
Vector Trigonometric Cosh (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

DEMA(*real*, *timeperiod*=?)

```
Double Exponential Moving Average (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

DIV(*real0*, *real1*)

```
Vector Arithmetic Div (Math Operators)

Inputs:
    real0: (any ndarray)
    real1: (any ndarray)
Outputs:
    real
```

DX(*high*, *low*, *close*, *timeperiod*=?)

```
Directional Movement Index (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    timeperiod: 14
Outputs:
    real
```

EMA(*real*, *timeperiod*=?)

```
Exponential Moving Average (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

EXP(*real*)

```
Vector Arithmetic Exp (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

FLOOR(*real*)

```
Vector Floor (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

HT_DCPERIOD(*real*)

```
Hilbert Transform - Dominant Cycle Period (Cycle Indicators)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

HT_DCPHASE(*real*)

```
Hilbert Transform - Dominant Cycle Phase (Cycle Indicators)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

HT_PHASOR(*real*)

```
Hilbert Transform - Phasor Components (Cycle Indicators)

Inputs:
    real: (any ndarray)
Outputs:
    inphase
    quadrature
```

HT_SINE(*real*)

```
Hilbert Transform - SineWave (Cycle Indicators)

Inputs:
    real: (any ndarray)
Outputs:
    sine
    leadsine
```

HT_TRENDLINE(*real*)

```
Hilbert Transform - Instantaneous Trendline (Overlap Studies)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

HT_TRENDMODE(*real*)

```
Hilbert Transform - Trend vs Cycle Mode (Cycle Indicators)

Inputs:
    real: (any ndarray)
Outputs:
    integer (values are -100, 0 or 100)
```

KAMA(*real*, *timeperiod*=?)

```
Kaufman Adaptive Moving Average (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

LINEARREG(*real*, *timeperiod*=?)

```
Linear Regression (Statistic Functions)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 14
Outputs:
    real
```

LINEARREG_ANGLE(*real*, *timeperiod*=?)

```
Linear Regression Angle (Statistic Functions)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 14
Outputs:
    real
```

LINEARREG_INTERCEPT(*real*, *timeperiod*=?)

```
Linear Regression Intercept (Statistic Functions)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 14
Outputs:
    real
```

LINEARREG_SLOPE(*real*, *timeperiod*=?)

```
Linear Regression Slope (Statistic Functions)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 14
Outputs:
    real
```

LN(*real*)

```
Vector Log Natural (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

LOG10(*real*)

```
Vector Log10 (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

MA(*real*, *timeperiod*=?, *matype*=?)

```
Moving average (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
    matype: 0 (Simple Moving Average)
Outputs:
    real
```

MACD(*real*, *fastperiod*=?, *slowperiod*=?, *signalperiod*=?)

```
Moving Average Convergence/Divergence (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    fastperiod: 12
    slowperiod: 26
    signalperiod: 9
Outputs:
    macd
    macdsignal
    macdhist
```

MACDEXT(*real*, *fastperiod*=?, *fastmatype*=?, *slowperiod*=?, *slowmatype*=?, *signalperiod*=?, *signalmatype*=?)

```
MACD with controllable MA type (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    fastperiod: 12
    fastmatype: 0
    slowperiod: 26
    slowmatype: 0
    signalperiod: 9
    signalmatype: 0
Outputs:
    macd
    macdsignal
    macdhist
```

MACDFIX(*real*, *signalperiod*=?)

```
Moving Average Convergence/Divergence Fix 12/26 (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    signalperiod: 9
Outputs:
    macd
    macdsignal
    macdhist
```

MAMA(*real*, *fastlimit*=?, *slowlimit*=?)

```
MESA Adaptive Moving Average (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    fastlimit: 0.5
    slowlimit: 0.05
Outputs:
    mama
    fama
```

MAVP(*real*, *periods*, *minperiod*=?, *maxperiod*=?, *matype*=?)

```
Moving average with variable period (Overlap Studies)

Inputs:
    real: (any ndarray)
    periods: (any ndarray)
Parameters:
    minperiod: 2
    maxperiod: 30
    matype: 0 (Simple Moving Average)
Outputs:
    real
```

MAX(*real*, *timeperiod*=?)

```
Highest value over a specified period (Math Operators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

MAXINDEX(*real*, *timeperiod*=?)

```
Index of highest value over a specified period (Math Operators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    integer (values are -100, 0 or 100)
```

MEDPRICE(*high*, *low*)

```
Median Price (Price Transform)

Inputs:
    prices: ['high', 'low']
Outputs:
    real
```

MFI(*high*, *low*, *close*, *volume*, *timeperiod*=?)

```
Money Flow Index (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close', 'volume']
Parameters:
    timeperiod: 14
Outputs:
    real
```

MIDPOINT(*real*, *timeperiod*=?)

```
MidPoint over period (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 14
Outputs:
    real
```

MIDPRICE(*high*, *low*, *timeperiod*=?)

```
Midpoint Price over period (Overlap Studies)

Inputs:
    prices: ['high', 'low']
Parameters:
    timeperiod: 14
Outputs:
    real
```

MIN(*real*, *timeperiod*=?)

```
Lowest value over a specified period (Math Operators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

MININDEX(*real*, *timeperiod*=?)

```
Index of lowest value over a specified period (Math Operators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    integer (values are -100, 0 or 100)
```

MINMAX(*real*, *timeperiod*=?)

```
Lowest and highest values over a specified period (Math Operators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    min
    max
```

MINMAXINDEX(*real*, *timeperiod*=?)

```
Indexes of lowest and highest values over a specified period (Math Operators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    minidx
    maxidx
```

MINUS_DI(*high*, *low*, *close*, *timeperiod*=?)

```
Minus Directional Indicator (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    timeperiod: 14
Outputs:
    real
```

MINUS_DM(*high*, *low*, *timeperiod*=?)

```
Minus Directional Movement (Momentum Indicators)

Inputs:
    prices: ['high', 'low']
Parameters:
    timeperiod: 14
Outputs:
    real
```

MOM(*real*, *timeperiod*=?)

```
Momentum (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 10
Outputs:
    real
```

MULT(*real0*, *real1*)

```
Vector Arithmetic Mult (Math Operators)

Inputs:
    real0: (any ndarray)
    real1: (any ndarray)
Outputs:
    real
```

NATR(*high*, *low*, *close*, *timeperiod*=?)

```
Normalized Average True Range (Volatility Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    timeperiod: 14
Outputs:
    real
```

OBV(*real*, *volume*)

```
On Balance Volume (Volume Indicators)

Inputs:
    real: (any ndarray)
    prices: ['volume']
Outputs:
    real
```

PLUS_DI(*high*, *low*, *close*, *timeperiod*=?)

```
Plus Directional Indicator (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    timeperiod: 14
Outputs:
    real
```

PLUS_DM(*high*, *low*, *timeperiod*=?)

```
Plus Directional Movement (Momentum Indicators)

Inputs:
    prices: ['high', 'low']
Parameters:
    timeperiod: 14
Outputs:
    real
```

PPO(*real*, *fastperiod*=?, *slowperiod*=?, *matype*=?)

```
Percentage Price Oscillator (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    fastperiod: 12
    slowperiod: 26
    matype: 0 (Simple Moving Average)
Outputs:
    real
```

ROC(*real*, *timeperiod*=?)

```
Rate of change : ((real/prevPrice)-1)*100 (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 10
Outputs:
    real
```

ROCP(*real*, *timeperiod*=?)

```
Rate of change Percentage: (real-prevPrice)/prevPrice (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 10
Outputs:
    real
```

ROCR(*real*, *timeperiod*=?)

```
Rate of change ratio: (real/prevPrice) (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 10
Outputs:
    real
```

ROCR100(*real*, *timeperiod*=?)

```
Rate of change ratio 100 scale: (real/prevPrice)*100 (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 10
Outputs:
    real
```

RSI(*real*, *timeperiod*=?)

```
Relative Strength Index (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 14
Outputs:
    real
```

SAR(*high*, *low*, *acceleration*=?, *maximum*=?)

```
Parabolic SAR (Overlap Studies)

Inputs:
    prices: ['high', 'low']
Parameters:
    acceleration: 0.02
    maximum: 0.2
Outputs:
    real
```

SAREXT(*high*, *low*, *startvalue*=?, *offsetonreverse*=?, *accelerationinitlong*=?, *accelerationlong*=?, *accelerationmaxlong*=?, *accelerationinitshort*=?, *accelerationshort*=?, *accelerationmaxshort*=?)

```
Parabolic SAR - Extended (Overlap Studies)

Inputs:
    prices: ['high', 'low']
Parameters:
    startvalue: 0
    offsetonreverse: 0
    accelerationinitlong: 0.02
    accelerationlong: 0.02
    accelerationmaxlong: 0.2
    accelerationinitshort: 0.02
    accelerationshort: 0.02
    accelerationmaxshort: 0.2
Outputs:
    real
```

SIN(*real*)

```
Vector Trigonometric Sin (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

SINH(*real*)

```
Vector Trigonometric Sinh (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

SMA(*real*, *timeperiod*=?)

```
Simple Moving Average (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

SQRT(*real*)

```
Vector Square Root (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

STDDEV(*real*, *timeperiod*=?, *nbdev*=?)

```
Standard Deviation (Statistic Functions)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 5
    nbdev: 1
Outputs:
    real
```

STOCH(*high*, *low*, *close*, *fastk_period*=?, *slowk_period*=?, *slowk_matype*=?, *slowd_period*=?, *slowd_matype*=?)

```
Stochastic (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    fastk_period: 5
    slowk_period: 3
    slowk_matype: 0
    slowd_period: 3
    slowd_matype: 0
Outputs:
    slowk
    slowd
```

STOCHF(*high*, *low*, *close*, *fastk_period*=?, *fastd_period*=?, *fastd_matype*=?)

```
Stochastic Fast (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    fastk_period: 5
    fastd_period: 3
    fastd_matype: 0
Outputs:
    fastk
    fastd
```

STOCHRSI(*real*, *timeperiod*=?, *fastk_period*=?, *fastd_period*=?, *fastd_matype*=?)

```
Stochastic Relative Strength Index (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 14
    fastk_period: 5
    fastd_period: 3
    fastd_matype: 0
Outputs:
    fastk
    fastd
```

SUB(*real0*, *real1*)

```
Vector Arithmetic Substraction (Math Operators)

Inputs:
    real0: (any ndarray)
    real1: (any ndarray)
Outputs:
    real
```

SUM(*real*, *timeperiod*=?)

```
Summation (Math Operators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

T3(*real*, *timeperiod*=?, *vfactor*=?)

```
Triple Exponential Moving Average (T3) (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 5
    vfactor: 0.7
Outputs:
    real
```

TAN(*real*)

```
Vector Trigonometric Tan (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

TANH(*real*)

```
Vector Trigonometric Tanh (Math Transform)

Inputs:
    real: (any ndarray)
Outputs:
    real
```

TEMA(*real*, *timeperiod*=?)

```
Triple Exponential Moving Average (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

TRANGE(*high*, *low*, *close*)

```
True Range (Volatility Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Outputs:
    real
```

TRIMA(*real*, *timeperiod*=?)

```
Triangular Moving Average (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

TRIX(*real*, *timeperiod*=?)

```
1-day Rate-Of-Change (ROC) of a Triple Smooth EMA (Momentum Indicators)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```

TSF(*real*, *timeperiod*=?)

```
Time Series Forecast (Statistic Functions)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 14
Outputs:
    real
```

TYPPRICE(*high*, *low*, *close*)

```
Typical Price (Price Transform)

Inputs:
    prices: ['high', 'low', 'close']
Outputs:
    real
```

ULTOSC(*high*, *low*, *close*, *timeperiod1*=?, *timeperiod2*=?, *timeperiod3*=?)

```
Ultimate Oscillator (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    timeperiod1: 7
    timeperiod2: 14
    timeperiod3: 28
Outputs:
    real
```

VAR(*real*, *timeperiod*=?, *nbdev*=?)

```
Variance (Statistic Functions)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 5
    nbdev: 1
Outputs:
    real
```

WCLPRICE(*high*, *low*, *close*)

```
Weighted Close Price (Price Transform)

Inputs:
    prices: ['high', 'low', 'close']
Outputs:
    real
```

WILLR(*high*, *low*, *close*, *timeperiod*=?)

```
Williams' %R (Momentum Indicators)

Inputs:
    prices: ['high', 'low', 'close']
Parameters:
    timeperiod: 14
Outputs:
    real
```

WMA(*real*, *timeperiod*=?)

```
Weighted Moving Average (Overlap Studies)

Inputs:
    real: (any ndarray)
Parameters:
    timeperiod: 30
Outputs:
    real
```