

# jhTAlib

Joost Hoeks

2019-03-12

## Contents

<b>jhTAlib</b>	<b>2</b>
Depends only on	2
Docs	2
Install	2
Update	3
Examples	3
Example 1	3
Example 2	3
Example 3	3
Example 4	4
Example 5	4
Example 6	4
Example 7	4
Example 8	4
Example 9	5
Example 10	5
Test	5
Reference	5
Behavioral Techniques	5
Cycle Indicators	8
Data	9
Event Driven	10
Experimental	10
General	12
Information	13
Math Functions	13
Momentum Indicators	18
Overlap Studies	22
Pattern Recognition	24
Price Transform	28
Statistic Functions	28
Volatility Indicators	31

Volume Indicators . . . . .	31
-----------------------------	----

## jhTAlib

Technical Analysis Library Time-Series

You can use and import it for your:

- Technical Analysis Software
- Charting Software
- Backtest Software
- Trading Robot Software
- Trading Software in general

Work in progress...

---

### Depends only on

- The Python Standard Library

---

### Docs

- [html](#)
- [pdf](#)
- [rst](#)

---

### Install

From PyPI:

```
$ [sudo] pip3 install jhtalib
```

From source:

```
$ git clone https://github.com/joosthoeks/jhTAlib.git
$ cd jhTAlib
$ [sudo] pip3 install -e .
```

---

## Update

From PyPI:

```
$ [sudo] pip3 install --upgrade jhtalib
```

From source:

```
$ cd jhTAlib  
$ git pull [upstream master]
```

---

## Examples

```
$ cd example/
```

### Example 1

```
$ python3 example-1-plot.py
```

or

[Open In Colab](#)

---

### Example 2

```
$ python3 example-2-plot.py
```

or

[Open In Colab](#)

---

### Example 3

```
$ python3 example-3-plot.py
```

or

[Open In Colab](#)

---

#### **Example 4**

```
$ python3 example-4-plot-quandl.py
```

or

[Open In Colab](#)

---

#### **Example 5**

```
$ python3 example-5-plot-quandl.py
```

or

[Open In Colab](#)

---

#### **Example 6**

```
$ python3 example-6-plot-quandl.py
```

or

[Open In Colab](#)

---

#### **Example 7**

```
$ python3 example-7-quandl-2-df.py
```

or

[Open In Colab](#)

---

#### **Example 8**

```
$ python3 example-8-alphavantage-2-df.py
```

or

[Open In Colab](#)

---

### Example 9

```
$ python3 example-9-cryptocompare-2-df.py
```

or

[Open In Colab](#)

---

### Example 10

DF NumPy Pandas

[Open In Colab](#)

---

### Test

```
$ cd test/  
$ python3 test.py
```

---

### Reference

```
import jhtalib as jhta
```

### Behavioral Techniques

#### ATH | All Time High | DONE

- dict of lists = jhta.ATH(df, price='High')
- 

#### LMC | Last Major Correction | DONE

- dict of lists = jhta.LMC(df, price='Low')
-

#### **PP | Pivot Point | DONE**

- dict of lists = jhta.PP(df)
- 

#### **FIBOPR | Fibonacci Price Retracements | DONE**

- dict of lists = jhta.FIBOPR(df, price='Close')
- 

#### **FIBTR | Fibonacci Time Retracements |**

- 
- 

#### **GANNPR | W. D. Gann Price Retracements | DONE**

- dict of lists = jhta.GANNPR(df, price='Close')
- 

#### **GANNTR | W. D. Gann Time Retracements |**

- 
- 

#### **JDN | Julian Day Number | DONE**

- jdn = jhta.JDN(utc\_year, utc\_month, utc\_day)
- 

#### **JD | Julian Date | DONE**

- jd = jhta.JD(utc\_year, utc\_month, utc\_day, utc\_hour, utc\_minute, utc\_second)
- 

#### **SUNC | Sun Cycle |**

- 
-

MERCURYC | Mercury Cycle |

•

---

VENUSC | Venus Cycle |

•

---

EARTH C | Earth Cycle |

•

---

MARSC | Mars Cycle |

•

---

JUPITERC | Jupiter Cycle |

•

---

SATURNC | Saturn Cycle |

•

---

URANUSC | Uranus Cycle |

•

---

NEPTUNEC | Neptune Cycle |

•

---

PLUTOOC | Pluto Cycle |

- 

---

MOONC | Moon Cycle |

- 

---

Cycle Indicators

HT\_DCPERIOD | Hilbert Transform - Dominant Cycle Period |

- 

---

HT\_DCPHASE | Hilbert Transform - Dominant Cycle Phase |

- 

---

HT\_PHASOR | Hilbert Transform - Phasor Components |

- 

---

HT\_SINE | Hilbert Transform - SineWave |

- 

---

HT\_TRENDLINE | Hilbert Transform - Instantaneous Trendline |

- 

---

HT\_TRENDMODE | Hilbert Transform - Trend vs Cycle Mode |

- 

---



#### **TS | Trend Score | DONE**

- `list = jhta.TS(df, n, price='Close')`
- 

#### **Data**

#### **CSV2DF | CSV file 2 DataFeed | DONE**

- `dict of tuples = jhta.CSV2DF(csv_file_path)`
- 

#### **CSVURL2DF | CSV file url 2 DataFeed | DONE**

- `dict of tuples = jhta.CSVURL2DF(csv_file_url)`
- 

#### **DF2CSV | DataFeed 2 CSV file | DONE**

- `csv file = jhta.DF2CSV(df, csv_file_path)`
- 

#### **DF2DFREV | DataFeed 2 DataFeed Reversed | DONE**

- `dict of tuples = jhta.DF2DFREV(df)`
- 

#### **DF2DFWIN | DataFeed 2 DataFeed Window | DONE**

- `dict of tuples = jhta.DF2DFWIN(df, start=0, end=10)`
- 

#### **DF\_HEAD | DataFeed HEAD | DONE**

- `dict of tuples = jhta.DF_HEAD(df, n=5)`
- 

#### **DF\_TAIL | DataFeed TAIL | DONE**

- `dict of tuples = jhta.DF_TAIL(df, n=5)`
-

**DF2HEIKIN\_\_ASHI | DataFeed 2 Heikin-Ashi DataFeed | DONE**

- dict of tuples = jhta.DF2HEIKIN\_\_ASHI(df)
- 

**Event Driven**

**ASI | Accumulation Swing Index (J. Welles Wilder) | DONE**

- list = jhta.ASI(df, L)
- 

**SI | Swing Index (J. Welles Wilder) | DONE**

- list = jhta.SI(df, L)
- 

**Experimental**

**JH\_SAVGP | Swing Average Price - previous Average Price | DONE**

- list = jhta.JH\_SAVGP(df)
- 

**JH\_SAVGPS | Swing Average Price - previous Average Price Sum-  
mation | DONE**

- list = jhta.JH\_SAVGPS(df)
- 

**JH\_SCO | Swing Close - Open | DONE**

- list = jhta.JH\_SCO(df)
- 

**JH\_SCOS | Swing Close - Open Summation | DONE**

- list = jhta.JH\_SCOS(df)
-

**JH\_SMEDP | Swing Median Price - previous Median Price | DONE**

- `list = jhta.JH_SMEDP(df)`
- 

**jh\_SMEDPS | Swing Median Price - previous Median Price Summation | DONE**

- `list = jhta.JH_SMEDPS(df)`
- 

**JH\_SPP | Swing Price - previous Price | DONE**

- `list = jhta.JH_SPP(df, price='Close')`
- 

**JH\_SPPS | Swing Price - previous Price Summation | DONE**

- `list = jhta.JH_SPPS(df, price='Close')`
- 

**JH\_STYPP | Swing Typical Price - previous Typical Price | DONE**

- `list = jhta.JH_STYPP(df)`
- 

**JH\_STYPPS | Swing Typical Price - previous Typical Price Summation | DONE**

- `list = jhta.JH_STYPPS(df)`
- 

**JH\_SWCLP | Swing Weighted Close Price - previous Weighted Close Price | DONE**

- `list = jhta.JH_SWCLP(df)`
-

**JH\_SWCLPS | Swing Weighted Close Price - previous Weighted Close Price Summation | DONE**

- `list = jhta.JH_SWCLPS(df)`
- 

**General**

**NORMALIZE | Normalize | DONE**

- `list = jhta.NORMALIZE(df, price_max='High', price_min='Low', price='Close')`
- 

**STANDARDIZE | Standardize | DONE**

- `list = jhta.STANDARDIZE(df, price='Close')`
- 

**SPREAD | Spread | DONE**

- `list = jhta.SPREAD(df1, df2, price1='Close', price2='Close')`
- 

**CP | Comparative Performance | DONE**

- `list = jhta.CP(df1, df2, price1='Close', price2='Close')`
- 

**CRSI | Comparative Relative Strength Index | DONE**

- `list = jhta.CRSI(df1, df2, n, price1='Close', price2='Close')`
- 

**CS | Comparative Strength | DONE**

- `list = jhta.CS(df1, df2, price1='Close', price2='Close')`
-

#### **HR | Hit Rate / Win Rate | DONE**

- `float = jhta.HR(hit_trades_int, total_trades_int)`
- 

#### **PLR | Profit/Loss Ratio | DONE**

- `float = jhta.PLR(mean_trade_profit_float, mean_trade_loss_float)`
- 

#### **EV | Expected Value | DONE**

- `float = jhta.EV(hittrade_float, mean_trade_profit_float, mean_trade_loss_float)`
- 

#### **POR | Probability of Ruin (Table of Lucas and LeBeau) | DONE**

- `int = jhta.POR(hittrade_float, profit_loss_ratio_float)`
- 

#### **Information**

##### **INFO | Print df Information | DONE**

- `print = jhta.INFO(df, price='Close')`
- 

##### **INFO\_TRADES | Print Trades Information | DONE**

- `print = jhta.INFO_TRADES(profit_trades_list, loss_trades_list)`
- 

#### **Math Functions**

##### **EXP | Exponential | DONE**

- `list = jhta.EXP(df, price='Close')`
-

#### LOG | Logarithm | DONE

- `list = jhta.LOG(df, price='Close')`
- 

#### LOG10 | Base-10 Logarithm | DONE

- `list = jhta.LOG10(df, price='Close')`
- 

#### SQRT | Square Root | DONE

- `list = jhta.SQRT(df, price='Close')`
- 

#### ACOS | Arc Cosine | DONE

- `list = jhta.ACOS(df, price='Close')`
- 

#### ASIN | Arc Sine | DONE

- `list = jhta.ASIN(df, price='Close')`
- 

#### ATAN | Arc Tangent | DONE

- `list = jhta.ATAN(df, price='Close')`
- 

#### COS | Cosine | DONE

- `list = jhta.COS(df, price='Close')`
- 

#### SIN | Sine | DONE

- `list = jhta.SIN(df, price='Close')`
-

#### **TAN | Tangent | DONE**

- `list = jhta.TAN(df, price='Close')`
- 

#### **ACOSH | Inverse Hyperbolic Cosine | DONE**

- `list = jhta.ACOSH(df, price='Close')`
- 

#### **ASINH | Inverse Hyperbolic Sine | DONE**

- `list = jhta.ASINH(df, price='Close')`
- 

#### **ATANH | Inverse Hyperbolic Tangent | DONE**

- `list = jhta.ATANH(df, price='Close')`
- 

#### **COSH | Hyperbolic Cosine | DONE**

- `list = jhta.COSH(df, price='Close')`
- 

#### **SINH | Hyperbolic Sine | DONE**

- `list = jhta.SINH(df, price='Close')`
- 

#### **TANH | Hyperbolic Tangent | DONE**

- `list = jhta.TANH(df, price='Close')`
- 

#### **PI | Mathematical constant PI | DONE**

- `float = jhta.PI()`
-

**E | Mathematical constant E | DONE**

- `float = jhta.E()`
- 

**TAU | Mathematical constant TAU | DONE**

- `float = jhta.TAU()`
- 

**PHI | Mathematical constant PHI | DONE**

- `float = jhta.PHI()`
- 

**CEIL | Ceiling | DONE**

- `list = jhta.CEIL(df, price='Close')`
- 

**FLOOR | Floor | DONE**

- `list = jhta.FLOOR(df, price='Close')`
- 

**DEGREES | Radians to Degrees | DONE**

- `list = jhta.DEGREES(df, price='Close')`
- 

**RADIANS | Degrees to Radians | DONE**

- `list = jhta.RADIANS(df, price='Close')`
- 

**ADD | Addition High + Low | DONE**

- `list = jhta.ADD(df)`
-



**DIV | Division High / Low | DONE**

- `list = jhta.DIV(df)`
- 

**MAX | Highest value over a specified period | DONE**

- `list = jhta.MAX(df, n, price='Close')`
- 

**MAXINDEX | Index of highest value over a specified period |**

- 
- 

**MIN | Lowest value over a specified period | DONE**

- `list = jhta.MIN(df, n, price='Close')`
- 

**MININDEX | Index of lowest value over a specified period |**

- 
- 

**MINMAX | Lowest and Highest values over a specified period |**

- 
- 

**MINMAXINDEX | Indexes of lowest and highest values over a specified period |**

- 
- 

**MULT | Multiply High \* Low | DONE**

- `list = jhta.MULT(df)`
-

### SUB | Subtraction High - Low | DONE

- `list = jhta.SUB(df)`
- 

### SUM | Summation | DONE

- `list = jhta.SUM(df, n, price='Close')`
- 

### Momentum Indicators

#### ADX | Average Directional Movement Index |

- 
- 

#### ADXN | Average Directional Movement Index Rating |

- 
- 

### APO | Absolute Price Oscillator | DONE

- `list = jhta.APO(df, n_fast, n_slow, price='Close')`
- 

### AROON | Aroon |

- 
- 

### AROONOSC | Aroon Oscillator |

- 
- 

### BOP | Balance Of Power |

- 
-

CCI | Commodity Channel Index |

- 

---

CMO | Chande Momentum Oscillator |

- 

---

DX | Directional Movement Index |

- 

---

IMI | Intraday Momentum Index | DONE

- `list = jhta.IMI(df)`

---

MACD | Moving Average Convergence/Divergence |

- 

---

MACDEXT | MACD with controllable MA type |

- 

---

MACDFIX | Moving Average Convergence/Divergence Fix 12/26 |

- 

---

MFI | Money Flow Index |

- 

---

MINUS\_DI | Minus Directional Indicator |

- 

MINUS\_DM | Minus Directional Movement |

- 

MOM | Momentum | DONE

- `list = jhta.MOM(df, n, price='Close')`

PLUS\_DI | Plus Directional Indicator |

- 

PLUS\_DM | Plus Directional Movement |

- 

PPO | Percentage Price Oscillator |

- 

ROC | Rate of Change | DONE

- `list = jhta.ROC(df, n, price='Close')`

ROCP | Rate of Change Percentage | DONE

- `list = jhta.ROCP(df, n, price='Close')`

### ROCR | Rate of Change Ratio | DONE

- `list = jhta.ROCR(df, n, price='Close')`
- 

### ROCR100 | Rate of Change Ratio 100 scale | DONE

- `list = jhta.ROCR100(df, n, price='Close')`
- 

### RSI | Relative Strength Index | DONE

- `list = jhta.RSI(df, n, price='Close')`
- 

### STOCH | Stochastic |

- 
- 

### STOCHF | Stochastic Fast |

- 
- 

### STOCHRSI | Stochastic Relative Strength Index |

- 
- 

### TRIX | 1-day Rate-Of-Change (ROC) of a Triple Smooth EMA |

- 
- 

### ULTOSC | Ultimate Oscillator |

- 
-

### **WILLR | Williams' %R | DONE**

- `list = jhta.WILLR(df, n)`
- 

### **Overlap Studies**

### **BBANDS | Bollinger Bands | DONE**

- `dict of lists = jhta.BBANDS(df, n, f=2)`
- 

### **BBANDW | Bollinger Band Width | DONE**

- `list = jhta.BBANDW(df, n, f=2)`
- 

### **DEMA | Double Exponential Moving Average |**

- 
- 

### **EMA | Exponential Moving Average |**

- 
- 

### **ENVP | Envelope Percent | DONE**

- `dict of lists = jhta.ENVP(df, pct=.01, price='Close')`
- 

### **KAMA | Kaufman Adaptive Moving Average |**

- 
- 

### **MA | Moving Average |**

- 
-

**MAMA | MESA Adaptive Moving Average |**

- 

**MAVP | Moving Average with Variable Period |**

- 

**MIDPOINT | MidPoint over period | DONE**

- `list = jhta.MIDPOINT(df, n, price='Close')`

**MIDPRICE | MidPoint Price over period | DONE**

- `list = jhta.MIDPRICE(df, n)`

**MMR | Mayer Multiple Ratio | DONE**

- `list = jhta.MMR(df, n=200, price='Close')`

**SAR | Parabolic SAR | DONE**

- `list = jhta.SAR(df, af_step=.02, af_max=.2)`

**SAREXT | Parabolic SAR - Extended |**

- 

**SMA | Simple Moving Average | DONE**

- `list = jhta.SMA(df, n, price='Close')`

T3 | Triple Exponential Moving Average (T3) |

- 

TEMA | Triple Exponential Moving Average |

- 

TRIMA | Triangular Moving Average | DONE

- `list = jhta.TRIMA(df, n, price='Close')`

WMA | Weighted Moving Average

- 

Pattern Recognition

CDL2CROWS | Two Crows |

CDL3BLACKCROWS | Three Black Crows |

CDL3INSIDE | Three Inside Up/Down |

CDL3LINESTRIKE | Three-Line Strike |

CDL3OUTSIDE | Three Outside Up/Down |

CDL3STARSINSOUTH | Three Stars In The South |

CDL3WHITESOLDIERS | Three Advancing White Soldiers |

CDLABANDONEDBABY | Abandoned Baby |



**CDLADVANCEBLOCK** | Advance Block |

**CDLBELTHOLD** | Belt-hold |

**CDLBREAKAWAY** | Breakaway |

**CDLCLOSINGMARUBOZU** | Closing Marubozu |

**CDLCONSEALBABYSWALL** | Concealing Baby Swallow |

**CDLCOUNTERATTACK** | Counterattack |

**CDLDARKCLOUDCOVER** | Dark Cloud Cover |

**CDLDOJI** | Doji |

**CDLDOJISTAR** | Doji Star |

**CDLDRAGONFLYDOJI** | Dragonfly Doji |

**CDLENGULFING** | Engulfing Pattern |

**CDLEVENINGDOJISTAR** | Evening Doji Star |

**CDLEVENINGSTAR** | Evening Star |

**CDLGAPSIDESIDEWHITE** | Up/Down-gap side-by-side white lines  
|

**CDLGRAVESTONEDOJI** | Gravestone Doji |

**CDLHAMMER** | Hammer |

**CDLHANGINGMAN** | Hanging Man |

CDLHARAMI | Harami Pattern |

CDLHARAMICROSS | Harami Cross Pattern |

CDLHIGHWAVE | High-Wave Candle |

CDLHIKKAKE | Hikkake Pattern |

CDLHIKKAKEMOD | Modified Hikkake Pattern |

CDLHOMINGPIGEON | Homing Pigeon |

CDLIDENTICAL3CROWS | Identical Three Crows |

CDLINNECK | In-Neck Pattern |

CDLINVERTEDHAMMER | Inverted Hammer |

CDLKICKING | Kicking |

CDLKICKINGBYLENGTH | Kicking - bull/bear determined by the longer marubozu |

CDLLADDERBOTTOM | Ladder Bottom |

CDLLONGLEGGEDDOJI | Long Legged Doji |

CDLLONGLINE | Long Line Candle |

CDLMARUBOZU | Marubozu |

CDLMATCHINGLOW | Matching Low |

CDLMATHOLD | Mat Hold |

CDLMORNINGDOJISTAR | Morning Doji Star |

CDLMORNINGSTAR | Morning Star |

CDLONNECK | On-Neck Pattern |

CDLPIERCING | Piercing Pattern |

CDLRICKSHAWMAN | Rickshaw Man |

CDLRISEFALL3METHODS | Rising/Falling Three Methods |

CDLSEPARATINGLINES | Separating Lines |

CDLSHOOTINGSTAR | Shooting Star |

CDLSHORTLINE | Short Line Candle |

CDLSPINNINGTOP | Spinning Top |

CDLSTALLEDPATTERN | Stalled Pattern |

CDLTICKSANDWICH | Stick Sandwich |

CDLTAKURI | Takuri (Dragonfly Doji with very long lower shadow)  
|

CDLTASUKIGAP | Tasuki Gap |

CDLTHRUSTING | Thrusting Pattern |

CDLTRISTAR | Tristar Pattern |

CDLUNIQUE3RIVER | Unique 3 River |

**CDLUPSIDEGAP2CROWS | Upside Gap Two Crows |**

**CDLXSIDEGAP3METHODS | Upside/Downside Gap Three Methods |**

**Price Transform**

**AVGPRICE | Average Price | DONE**

- `list = jhta.AVGPRICE(df)`
- 

**MEDPRICE | Median Price | DONE**

- `list = jhta.MEDPRICE(df)`
- 

**TYPPRICE | Typical Price | DONE**

- `list = jhta.TYPPRICE(df)`
- 

**WCLPRICE | Weighted Close Price | DONE**

- `list = jhta.WCLPRICE(df)`
- 

**Statistic Functions**

**MEAN | Arithmetic mean (average) of data | DONE**

- `list = jhta.MEAN(df, n, price='Close')`
- 

**HARMONIC\_MEAN | Harmonic mean of data | DONE**

- `list = jhta.HARMONIC_MEAN(df, n, price='Close')`
-

**MEDIAN | Median (middle value) of data | DONE**

- `list = jhta.MEDIAN(df, n, price='Close')`
- 

**MEDIAN\_LOW | Low median of data | DONE**

- `list = jhta.MEDIAN_LOW(df, n, price='Close')`
- 

**MEDIAN\_HIGH | High median of data | DONE**

- `list = jhta.MEDIAN_HIGH(df, n, price='Close')`
- 

**MEDIAN\_GROUPED | Median, or 50th percentile, of grouped data | DONE**

- `list = jhta.MEDIAN_GROUPED(df, n, price='Close', interval=1)`
- 

**MODE | Mode (most common value) of discrete data | DONE**

- `list = jhta.MODE(df, n, price='Close')`
- 

**PSTDEV | Population standard deviation of data | DONE**

- `list = jhta.PSTDEV(df, n, price='Close', mu=None)`
- 

**PVARIANCE | Population variance of data | DONE**

- `list = jhta.PVARIANCE(df, n, price='Close', mu=None)`
- 

**STDEV | Sample standard deviation of data | DONE**

- `list = jhta.STDEV(df, n, price='Close', xbar=None)`
-

#### **VARIANCE | Sample variance of data | DONE**

- `list = jhta.VARIANCE(df, n, price='Close', xbar=None)`
- 

#### **COV | Covariance | DONE**

- `float = jhta.COV(list1, list2)`
- 

#### **COVARIANCE | Covariance | DONE**

- `list = jhta.COVARANCE(df1, df2, n, price1='Close', price2='Close')`
- 

#### **COR | Correlation | DONE**

- `float = jhta.COR(list1, list2)`
- 

#### **CORRELATION | Correlation | DONE**

- `list = jhta.CORRELATION(df1, df2, n, price1='Close', price2='Close')`
- 

#### **PCOR | Population Correlation | DONE**

- `float = jhta.PCOR(list1, list2)`
- 

#### **PCORRELATION | Population Correlation | DONE**

- `list = jhta.PCORRELATION(df1, df2, n, price1='Close', price2='Close')`
- 

#### **BETA | Beta | DONE**

- `float = jhta.BETA(list1, list2)`
-

### **BETAS | Betas | DONE**

- `list = jhta.BETAS(df1, df2, n, price1='Close', price2='Close')`
- 

### **LSR | Least Squares Regression | DONE**

- `list = jhta.LSR(df, price='Close', predictions_int=0)`
- 

### **SLR | Simple Linear Regression | DONE**

- `list = jhta.SLR(df, price='Close', predictions_int=0)`
- 

### **Volatility Indicators**

#### **ATR | Average True Range | DONE**

- `list = jhta.ATR(df, n)`
- 

#### **NATR | Normalized Average True Range |**

- 
- 

#### **TRANGE | True Range | DONE**

- `list = jhta.TRANGE(df)`
- 

### **Volume Indicators**

#### **AD | Chaikin A/D Line | DONE**

- `list = jhta.AD(df)`
-

**ADOSC | Chaikin A/D Oscillator |**

- 
- 

**OBV | On Balance Volume | DONE**

- `list = jhta.OBV(df)`
-