A Universal Stock Screen Application

Copyright © 2019 Jonathan Kinlay (jonathan@jonathankinlay.com)

Requirement

The requirement was to build a stock screening application capable of filtering worldwide stocks on the following parameters:

- The current stock price is above both the 150-day (30-week) and the 200-day (40-week) moving average price lines.
- The 150-day moving average is above the 200-day moving average.
- The 200-day moving average line is trending up for at least 1 month (preferably 4–5 months minimum in most cases).
- The 50-day (10-week) moving average is above both the 150-day and 200-day moving averages.
- The current stock price is trading above the 50-day moving average.
- The current stock price is at least 30 percent above its 52-week low. (Many of the best selections will be 100 percent, 300 percent, or greater above their 52-week low before they emerge from a solid consolidation period and mount a large scale advance.)
- The current stock price is within at least 25 percent of its 52-week high (the closer to a new high the better).
- EPS YOY growth +10% (last 3 years)
- Revenue YOY growth +10% (last 3 years)
- Net Profit Margin YOY growth +10% (last 3 years)

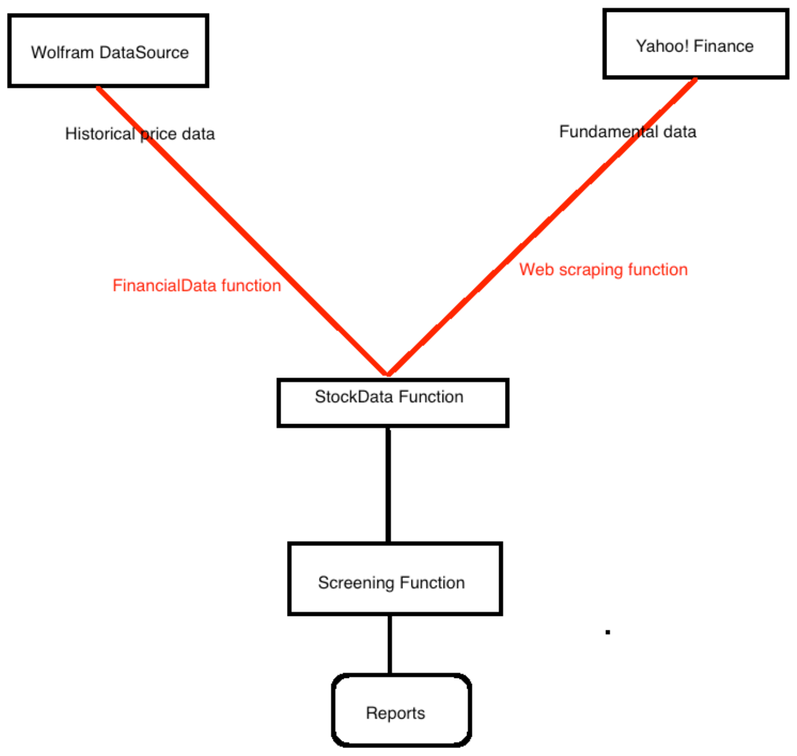
System Architecture

The application was developed in Mathematica and the Wolfram Language. This is a good choice for rapid prototyping of small applications, as it offers the advantages of speed of development, a convenient user interface and access to online data sources. More specifically:

- The Mathematica front-end provides a convenient and comfortable user interface for small-scale applications.
- Mathematica offers access to Wolfram data sources, including stock data, via its FinancialData function and the Wolfram Alpha platform.
- It is straightforward to develop web-scraping applications in the Wolfram Language, in order to be able to retrieve data from the Yahoo Finance and other web sites.
- WL provides extensive computational capabilities for data manipulation and analysis that require only limited additional programming to meet the specification.

- Although a scripting language such as WL is slower in execution speed than compiled languages like C++ and Java, this is not a major concern for this application as it would be for, say, a low-latency high frequency trading system.
- On the other hand, WL offers capabilities to parallelize the code very easily, making use of available multi-core processors to accelerate execution.
- The flexibility of the language makes it straightforward to generalize the application to provide add-on functionality.

The structure of the application is as fol-
lows:



Data Sources

Wolfram provides a reliable source of high quality historical stock data for both US and international exchanges. The data can be downloaded efficiently into the Mathematica application using the WL FinancialData function and manipulated to provide the required moving average and other time series. The advantage of using Wolfram as the source of historical data is that firstly, there are no api’s to navigate and secondly, it becomes trivially easy to parameterize the screening filters, for example to select moving averages of different lengths from those originally specified.

As the breadth of fundamental data offered by Wolfram is insufficient for this application, a second data source, the Yahoo! Finance platform, is accessed to obtain the required historical fundamental data. This requires a web scraping function to access and retrieve the data of interest from the Yahoo web site, which is straightforward to develop in the Wolfram Language.

It is worth noting that Mathematica/WL offers facilities to access data from other commercial providers, including Bloomberg, as well as trading platforms such as Interactive brokers, via the C++ api.

Program Structure

It is clearly desirable to separate the data retrieval and screening functions so that the two components may be re-engineered and/or plugged into other applications, as required. The web scraping function retrieves more of the data available from the Yahoo Finance platform than is required for this application, making it straightforward to extend the capabilities of the screening function.

The screening function is parameterized, with default values as specified in the requirements. So if the user wishes to relax some of the selection criteria (for example, to require that the stock is within,say, 50% of the 52-week high), it is trivial to make the necessary adjustments to the function inputs.

Extensions and Further Development

There are many ways in which this implementation could be enhanced:

- Access to other high quality market and fundamental data sources (e.g. Bloomberg, IB)
- Re-development to speed up some core components of the code using compilation
- Additional error-trapping and event handling
- Design of a custom user interface to facilitate the rapid deployment of new filters
- Multi-currency reporting

All of these extension can readily be accommodated in the Wolfram language and incorporated into the application architecture.

Limitations and Assumptions

Given the nature of the assignment a number of assumptions have been made and limitations to this initial version of the application:

- No attempt has been made to cross-check and validate the market and fundamental data against other sources.

- The error trapping in the code is limited and an interruption to the filtering process could occur if untrapped errors occur.
- Changes to the Yahoo Finance platform or api will likely necessitate redevelopment of the web scraping functionality.
- The WL code has not been optimized for speed, nor has any attempt been made to speed up the execution of the application through compilation, or the use of DLLs.
- It is assumed that the analysis will be updated periodically, perhaps monthly or, at most, on a weekly basis. If it becomes important to update more frequently (say, daily) then attention should be paid to address possible methods for enhancing the rate of execution.
- It is assumed that the standard Mathematica notebook interface will suffice for this initial release of the application.
- No attempt has been made to take account of exchange rates affecting non-US markets.
- The quality of the market and fundamental data is unlikely to be equivalent to that available from commercial providers.

Unit Testing

We apply the StockData function to download fundamental and technical data to be used for screening purposes on a test stock:

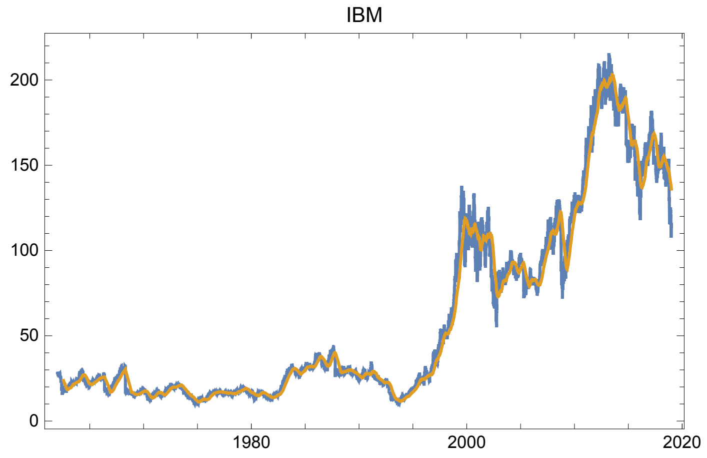
```
In[ ]:= {tsClose, tsMAFast, tsMAMedium, tsMASlow, ClosePrice, Summary, Stats, Valuation,
    TradingInfo, FinancialHighlights, ShareStatistics, DividendInfo, MAFast,
    MAMedium, MASlow, MASlowLag1M , Low52Week, High52Week, Financials,
    Revenue, RevenueGrowth3yr, NetIncome, NetProfitMargin, NPMGrowth3yr,
    Analysis, EarningsEstimates, RevenueEstimates, EarningsHistory, EPSTrend,
    EPSRevisions, GrowthEstimates, EPS, EPSGrowth3yr} = StockData["NYSE:IBM"];
```

The function returns the value of all of the test criteria variables used for screening, as well as additional information retrieved from Yahoo! Finance, organized by section, that may be useful in further analysis.

The function also returns time series in the stock closing prices and also in the fast, medium and slow moving average series that are set to the specified lengths of 50, 150 and 200 days. However, the user can easily adjust any of the parameter values to select other moving average lengths, as required.

```
In[∘]:= DateListPlot[{tsClose, tsMASlow}, PlotLabel → "IBM"]
```

Out[∘]=



The latest closing price and moving average values are as follows:

```
In[∘]:= {ClosePrice, MAFast, MAMedium, MASlow, MASlowLag1M}

Out[∘]= {115.21, 118.407, 133.58, 136.068, 140.042}
```

Financial Data Check

The downloaded financial information can easily be cross-checked against the data reported on the Yahoo! Finance site:

```
In[∘]:= TableForm[Join[{Revenue, NetIncome, NetProfitMargin, EPS}, 2],
         TableHeadings → {{"Revenue", "Net Income", "Net Profit Margin", "EPS (2017-2014)"},
           {2018, 2017, 2016, 2015}}]
```

Out[∘]//TableForm=

	2018	2017	2016	2015
Revenue	7.9139×10^{7}	7.9919×10^{7}	8.1741×10^{7}	9.2793×10^{7}
Net Income	5.753×10^{6}	1.1872×10^{7}	1.319×10^{7}	1.2022×10^{7}
Net Profit Margin	0.0726949	0.14855	0.161363	0.129557
EPS (2017-2014)	5.14	2.45	3.08	3.42

Check Net Income Calculations

```
In[∘]:= NetIncome / Revenue

Out[∘]= {0.0726949, 0.14855, 0.161363, 0.129557}
```

Check Growth Rates (3-year annually compounded)

```
In[∘]:= (Revenue[[1]] / Revenue[[3]]) ^ (1 / 3.0) - 1

Out[∘]= -0.0107254

In[∘]:= RevenueGrowth3yr

Out[∘]= -0.0107254

In[∘]:= (EPS[[1]] / EPS[[3]]) ^ (1 / 3.0) - 1

Out[∘]= 0.186144
```

```
In[•]:= EPSGrowth3yr
Out[•]= 0.186144

In[•]:= (NetProfitMargin[[1]] / NetProfitMargin[[3]]) ^ (1 / 3.0) - 1
Out[•]= -0.233404

In[•]:= NPMGrowth3yr
Out[•]= -0.233404
```

Screening Test

The StockCriteria function evaluates the stock on each of the ten performance criteria and returns and overall pass/fail result.

```
In[•]:= {boolPass, boolCondition} = StockCriteria[ClosePrice, MAFast, MAMedium, MASlow,
           MASlowLag1M, High52Week, Low52Week, EPSGrowth3yr, RevenueGrowth3yr, NPMGrowth3yr];
```

The variable boolCondition shows the result of each test criterion:

```
In[•]:= boolCondition
Out[•]= {False, False, False, False, False, False, False, True, False, False}
```

The variable boolPass shows the overall result:

```
In[•]:= boolPass
Out[•]= False
```

The results can be viewed in tabular format:

```
In[•]:= PrintTestResult
Out[•]//TableForm=
        Overall Result                                                     |False
        Current price is above 150 day MA and 200 day moving averages      |False
        150-day moving average is above the 200-day moving average         |False
        200-day moving average line is trending up for at least 1 month    |False
        50-day moving average is above both the 150-day and 200-day moving averages |False
        Current stock price is trading above the 50-day moving average     |False
        Current stock price is at least 30 percent above its 52-week low   |False
        Current stock price is within at least 25 percent of its 52-week high |False
        EPS YOY growth +10% (last 3 years)                                 |True
        Revenue YOY growth +10% (last 3 years)                             |False
        Net Profit Margin YOY growth +10% (last 3 years)                   |False
```

Stock Screening

Symbology

The Wolfram FinancialData function provides a complete list of stock symbols listed for a specified exchange. For example, for the LSE there are over 8,000 listed securities:

```
In[•]:= Length[symbolsL = FinancialData["L:*"]]
Out[•]= 8239
```

```
symbolsL[[RandomInteger[Length@symbolsL, 10]]]
```

```
Out[•]= symbols[L[{3361, 7950, 4800, 5599, 5845, 1832, 7213, 8044, 3394, 4262}]]
```

Symbols need to be converted to a format recognized by Yahoo! Finance, using the YahooSymbols function. For example:

```
In[•]:= YahooSymbol[symbolsL[[RandomInteger[Length@symbolsL, 10]]]]
```

```
Out[•]= {DNA.L, 0E3C.L, 0QF8.L, AGGG.L, 0RFY.L, LEED.L, GTC.L, OCPA.L, RUSB.L, 0IZ2.L}
```

Stock Screen: UK (LSE)

We make a random selection of 1,000 LSE stocks for screening:

```
In[•]:= selectedStocksUK = symbolsL[[RandomInteger[Length@symbolsL, 1000]]];
```

```
In[•]:= selectedStocksUK[[1 ;; 10]]
```

```
Out[•]= {L:0FTH, L:ALS, L:SBRE, L:0R3Y, L:UC85, L:0JHD, L:SUES, L:LLD5, L:ENDV, L:SUJA}
```

```
In[•]:= FinancialData[#, "Company"] & /@ selectedStocksUK[[1 ;; 10]] /.
         Missing["NotAvailable"] → Nothing
```

```
Out[•]= {Altus Strategies, Sabre Insurance Group, Lloyds Banking Group, Endeavour International}
```

```
Parallelize[result = Quiet[StockScreen[#] & /@ selectedStocksUK];];
screenedStocksUK = selectedStocksUK[[Flatten@Position[result[[ ;; , 1]], True]]]
```

Stock Screen: USA (NYSE, NASDAQ)

```
In[•]:= Length[symbolsNYSE = FinancialData["NYSE:*"]]
        Length[symbolsNASDAQ = FinancialData["NASDAQ:*"]]
```

```
Out[•]= 5025
```

```
Out[•]= 3253
```

We run the screen for a random selection of 1,000 symbols for the NYSE exchange and show the results

```
In[•]:= selectedStocksNYSE = symbolsNYSE[[RandomInteger[Length@symbolsNYSE, 1000]]];
        selectedStocksNYSE[[1 ;; 10]]
        FinancialData[#, "Company"] & /@ selectedStocksNYSE[[1 ;; 10]] /.
         Missing["NotAvailable"] → Nothing
```

```
Out[•]= {NYSE:BIV, NYSE:JCP, NYSE:GDV-PA, NYSE:AQN-U,
         NYSE:ULBR, NYSE:EWG, NYSE:JE-PA, NYSE:UWT, NYSE:ASHR, NYSE:ENR}
```

```
Out[•]= {JC Penney Co, Energizer Holdings}
```

```
Parallelize[result = Quiet[StockScreen[#] & /@ selectedStocksNYSE];];
screenedStocksNYSE = selectedStocksNYSE[[Flatten@Position[result[[ ;; , 1]], True]]]
```

Stock Screen: Germany (Frankfurt)

```
In[∘]:= Length[symbolsF = FinancialData["F:*"]]

Out[∘]= 16 044

In[∘]:= selectedStocksF = symbolsDE[[RandomInteger[Length@symbolsF, 1000]]];
        selectedStocksF[[1 ;; 10]]
        FinancialData[#, "Company"] & /@ selectedStocksF[[1 ;; 10]] /.
         Missing["NotAvailable"] → Nothing

Out[∘]= {F:3M0, F:PI1A, F:XMFA, F:UI4B, F:TIE, F:C6O, F:CE8G, F:UO1G, F:HO9, F:B5R}

Out[∘]= {Meiji Holdings Co, Pinetree Capital, Sumitomo Mitsui Financial,
         Taiheiyo Cement, Clovis Oncology, HNI, Berkeley Energia}

        Parallelize[result = Quiet[StockScreen[#] & /@ selectedStocksF];];
        screenedStocksDE = selectedStocksF[[Flatten@Position[result[[ ;; , 1]], True]]]
```

Stock Screen: Hong Kong

```
In[∘]:= Length[symbolsHK = FinancialData["HK:*"]]

Out[∘]= 2309

In[∘]:= selectedStocksHK = symbolsHK[[RandomInteger[Length@symbolsHK, 1000]]];
        selectedStocksHK[[1 ;; 10]]
        FinancialData[#, "Company"] & /@ selectedStocksHK[[1 ;; 10]] /.
         Missing["NotAvailable"] → Nothing

Out[∘]= {HK:01303, HK:00659, HK:02779, HK:00373,
         HK:01200, HK:00565, HK:08228, HK:00581, HK:01466, HK:00028}

Out[∘]= {Huili Resources (Group), NWS Holdings, China Xinhua Education Gr,
         Allied Group, Midland Holdings, Art Group Holdings, National Arts Enter,
         China Oriental Grp Co, Affluent Partners Hldgs, Tian An China Investment}

        Parallelize[result = Quiet[StockScreen[#] & /@ selectedStocksHK];];
        screenedStocksDE = selectedStocksHK[[Flatten@Position[result[[ ;; , 1]], True]]]
```

Stock Screen: India (Mumbai)

```
In[∘]:= Length[symbolsMU = FinancialData["MU:*"]]

Out[∘]= 9464
```

```
In[∘]:= selectedStocksMU = symbolsMU[[RandomInteger[Length@symbolsMU, 1000]]];
        selectedStocksMU[[1 ;; 10]]
        FinancialData[#, "Company"] & /@ selectedStocksMU[[1 ;; 10]] /.
         Missing["NotAvailable"] → Nothing

Out[∘]= {MU:ZPJP, MU:SAOA, MU:REJA, MU:HMT, MU:DX2J, MU:XZGD, MU:VOW4, MU:8NE2, MU:HG4U, MU:GSWE}

Out[∘]= {Sasol, Reply, Host Hotels & Resorts, Volkswagen, Caladrius Biosciences}

        Parallelize[result = Quiet[StockScreen[#] & /@ selectedStocksHK];];
        screenedStocksDE = selectedStocksHK[[Flatten@Position[result[[ ;; , 1]], True]]]
```

Stock Screen: Singapore

```
In[∘]:= Length[symbolsSI = FinancialData["SI:*"]]

Out[∘]= 898

In[∘]:= selectedStocksSI = symbolsSI[[RandomInteger[Length@symbolsSI, 1000]]];
        selectedStocksSI[[1 ;; 10]]
        FinancialData[#, "Company"] & /@ selectedStocksSI[[1 ;; 10]] /.
         Missing["NotAvailable"] → Nothing

Out[∘]= {SI:40N, SI:P5P, SI:42L, SI:K3HD, SI:5JS, SI:5TN, SI:R14, SI:H78, SI:5RC, SI:BQM}

Out[∘]= {Versalink Holdings, TLV Holdings, Aluminum Corp of China,
         Indofood Agri Resources, IEV Holdings, Ramba Energy,
         Hongkong Land Holdings, ES Group (Holdings), Tiong Woon Corp Holding}

        Parallelize[result = Quiet[StockScreen[#] & /@ selectedStocksHK];];
        screenedStocksDE = selectedStocksHK[[Flatten@Position[result[[ ;; , 1]], True]]]
```

Wolfram Language Code

```
StockData[ticker_, MALenFast_: 50, MALenMedium_: 150, MALenSlow_: 200] :=
 Module[{yahooticker, querystring, tsClose, tsMAFast, tsMAMedium, tsMASlow,
   MAFast, MAMedium, MASlow, MASlowLag1M, ClosePrice, Summary, Stats,
   Valuation, TradingInfo, FinancialHighlights, ShareStatistics, DividendInfo,
   Low52Week, High52Week, Financials, Revenue, RevenueGrowth3yr, NetIncome,
   NetProfitMargin, NPMGrowth3yr, Analysis, EarningsEstimates, RevenueEstimates,
   EarningsHistory, EPSTrend, EPSRevisions, GrowthEstimates, EPS, EPSGrowth3yr},
  (*
      ===============================================================
      ===============================================================
        StockData function version 1_0  Jan 2019
         Used to dowload historical market and fundamental data
          from Wolfram Research and Yahoo! Finance
      ===============================================================
      ===============================================================
  *)

  (*
     ===============================================================
      Download stock price data from Wolfram Research
      and calculate Moving Average Series
    ============================================================================
*)
yahooticker = YahooSymbol[ticker];
tsClose = TimeSeries[FinancialData[ticker, "Close", All]];
ClosePrice = Last@tsClose["Values"];
tsMAFast = MovingMap[Mean, tsClose, Quantity[MALenFast, "Days"]];
tsMAMedium = MovingMap[Mean, tsClose, Quantity[MALenMedium, "Days"]];
tsMASlow = MovingMap[Mean, tsClose, Quantity[MALenSlow, "Days"]];
MAFast = Last@tsMAFast["Values"];
MAMedium = Last@tsMAMedium["Values"];
MASlow = Last@tsMASlow["Values"];
MASlowLag1M =
 First@tsMASlow["SliceData", DatePlus[Last@tsMASlow["Dates"], {-1, "Month"}]];
(*
    ============================================================================
     Download statistics information from Yahoo! Finance and extract
     Valuation, Trading, Financial, and Dividend Info
    ============================================================================
*)

querystring =
 StringJoin["http://finance.yahoo.com/q/ks?s=", yahooticker, "+Key+Statistics"];
Stats = Last[Import[querystring, "Data"][[2]]];
If[Dimensions[Stats[[1]]] === {},
 Valuation = {};
 FinancialHighlights = {};,
 Valuation = Stats[[1, 1]];
 FinancialHighlights = Stats[[1, 2]];];
If[Dimensions[Stats[[2]]] === {},
 TradingInfo = {};
 High52Week = "N/A";
 Low52Week = "N/A";
 ShareStatistics = {};
 DividendInfo = {},
 TradingInfo = Stats[[2, 1]];
 High52Week = TradingInfo[[4, 2]];
 Low52Week = TradingInfo[[5, 2]];
 ShareStatistics = Stats[[2, 2]];
 DividendInfo = Stats[[2, 3]];];


(*
    ============================================================================
     Download financial statement summary from Yahoo! Finance and
     extract Revenues, Net Income information and compute growth rates
    ============================================================================
*)

querystring =
 StringJoin["http://finance.yahoo.com/quote/", yahooticker, "/financials"];
Financials = Last@Import[querystring, "Data"][[2]];
If[Dimensions[Financials[[2]]] === {} || Dimensions[Stats[[1]]] === {},
 Revenue = Table["N/A", 4];
```

```
     RevenueGrowth3yr = "N/A";,
     Revenue = Financials[[2, 2 ;;]];
     Revenue = Internal`StringToDouble[#] & /@ Revenue;
     RevenueGrowth3yr = (Revenue[[1]] / Revenue[[3]]) ^ (1 / 3.0) - 1;];
    If[Head[RevenueGrowth3yr] == Complex, RevenueGrowth3yr = -1];
    If[Dimensions[Financials[[-1]]] === {} || Dimensions[Stats[[1]]] === {},
     NetIncome = Table["N/A", 4];
     NetProfitMargin = Table["N/A", 4];
     NPMGrowth3yr = "N/A";,
     NetIncome = Financials[[-1, 2 ;;]];
     NetIncome = Internal`StringToDouble[#] & /@ NetIncome;
     NetProfitMargin = NetIncome / Revenue // N;
     NPMGrowth3yr = (NetProfitMargin[[1]] / NetProfitMargin[[3]]) ^ (1 / 3.0) - 1;];
    If[Head[NPMGrowth3yr] == Complex, NPMGrowth3yr = -1];
    (*
       ==========================================================================
        Download analysis section from Yahoo! Finance and
        EPS information and compute growth rates
        ==========================================================================
    *)
    querystring =
     StringJoin["http://finance.yahoo.com/quote/", yahooticker, "/analysis"];
    Analysis = Last@Import[querystring, "Data"][[2]];
    If[Dimensions@Analysis === {} || Dimensions[Stats[[1]]] === {},
     EPS = Table["N/A", 4];
     EPSGrowth3yr = "N/A";,
     EarningsEstimates = Analysis[[1]];
     RevenueEstimates = Analysis[[2]];
     EarningsHistory = Analysis[[3]];
     EPSTrend = Analysis[[4]];
     EPSRevisions = Analysis[[5]];
     GrowthEstimates = Analysis[[6]];
     EPS = EarningsHistory[[2]][[2, 2 ;;]];
     EPSGrowth3yr = (EPS[[1]] / EPS[[3]]) ^ (1 / 3.0) - 1;];
    If[Head[EPSGrowth3yr] == Complex, EPSGrowth3yr = -1];
    (*
       ==========================================================================
        Return results and close function
        ==========================================================================
    *)
    {tsClose, tsMAFast, tsMAMedium, tsMASlow, ClosePrice, Summary, Stats, Valuation,
     TradingInfo, FinancialHighlights, ShareStatistics, DividendInfo, MAFast,
     MAMedium, MASlow, MASlowLag1M , Low52Week, High52Week, Financials,
     Revenue, RevenueGrowth3yr, NetIncome, NetProfitMargin, NPMGrowth3yr,
     Analysis, EarningsEstimates, RevenueEstimates, EarningsHistory,
     EPSTrend, EPSRevisions, GrowthEstimates, EPS, EPSGrowth3yr}]


StockCriteria[CurrentPrice_, MAFastDay_, MAMediumDay_, MASlowDay_,
    MASlowDayLag1M_, High52Week_, Low52Week_, EPSGrowth3yr_, RevenueGrowth3yr_,
    NPMGrowth3yr_, PctoverLow_: 0.30, PctunderHigh_: 0.25, EPSGrowthRate_: 0.1,
    RevenueGrowthRate_: 0.1, NPMGrowthRate_: 0.1] := Module[{boolCondition, boolPass},
    (*
        ==========================================================================
        ==========================================================================
       StockCriteria function version 1_0  Jan 2019
       Set up the criteria to be used for screening:
      The current stock price is above both the 150-
       day (30-week) and the 200-day (40-week) moving average price lines.
      The 150-day moving average is above the 200-day moving average.
      The 200-day moving average line is trending up for at
         least 1 month (preferably 4-5 months minimum in most cases).
      The 50-day (10-week) moving average is above both the 150-
       day and 200-day moving averages.
      The current stock price is trading above the 50-day moving average.
      The current stock price is at least 30 percent above its 52-
       week low.(Many of the best selections will be 100 percent,
          300 percent or greater above their 52-week low before they emerge
            from a solid consolidation period and mount a large scale advance.)
      The current stock price is within at least 25 percent of its 52-
       week high (the closer to a new high the better).
      EPS YOY growth+10% (last 3 years)
      Revenue YOY growth+10% (last 3 years)
      Net Profit Margin YOY growth+10% (last 3 years)
       ===============================================================
     ===============================================================
*)
boolCondition = Table[{}, 10];
boolCondition[[1]] = CurrentPrice > MAMediumDay && CurrentPrice > MASlowDay;
boolCondition[[2]] = MAMediumDay > MASlowDay;
boolCondition[[3]] = MASlowDay > MASlowDayLag1M;
boolCondition[[4]] = MAFastDay > MAMediumDay && MAFastDay > MASlowDay;
boolCondition[[5]] = CurrentPrice > MAFastDay;
If[Head[Low52Week] === Real,
 boolCondition[[6]] = CurrentPrice ≥ (1 + PctoverLow) * Low52Week;,
 boolCondition[[6]] = False;];
If[Head[High52Week] === Real,
 boolCondition[[7]] = CurrentPrice ≥ (1 - PctunderHigh) * High52Week;,
 boolCondition[[7]] = False;];
If[Head[EPSGrowth3yr] === Real,
 boolCondition[[8]] = EPSGrowth3yr >= EPSGrowthRate;,
 boolCondition[[8]] = False;];
If[Head[RevenueGrowth3yr] === Real,
 boolCondition[[9]] = RevenueGrowth3yr >= RevenueGrowthRate;,
 boolCondition[[9]] = False;];
If[Head[NPMGrowth3yr] === Real,
 boolCondition[[10]] = NPMGrowth3yr >= NPMGrowthRate;,
 boolCondition[[10]] = False;];

boolPass = Fold[And, True, boolCondition];

{boolPass, boolCondition}]
```

```
In[∘]:= StockScreen[ticker_] :=
         Module[{tsClose, tsMAFast, tsMAMedium, tsMASlow, ClosePrice, Summary, Stats,
            Valuation, TradingInfo, FinancialHighlights, ShareStatistics, DividendInfo,
            MAFast, MAMedium, MASlow, MASlowLag1M , Low52Week, High52Week, Financials,
            Revenue, RevenueGrowth3yr, NetIncome, NetProfitMargin, NPMGrowth3yr,
            Analysis, EarningsEstimates, RevenueEstimates, EarningsHistory, EPSTrend,
            EPSRevisions, GrowthEstimates, EPS, EPSGrowth3yr, boolPass, boolCondition},
          {tsClose, tsMAFast, tsMAMedium, tsMASlow, ClosePrice, Summary, Stats, Valuation,
             TradingInfo, FinancialHighlights, ShareStatistics, DividendInfo, MAFast,
             MAMedium, MASlow, MASlowLag1M , Low52Week, High52Week, Financials,
             Revenue, RevenueGrowth3yr, NetIncome, NetProfitMargin, NPMGrowth3yr,
             Analysis, EarningsEstimates, RevenueEstimates, EarningsHistory, EPSTrend,
             EPSRevisions, GrowthEstimates, EPS, EPSGrowth3yr} = StockData[ticker];

          {boolPass, boolCondition} =
            StockCriteria[ClosePrice, MAFast, MAMedium, MASlow, MASlowLag1M,
             High52Week, Low52Week, EPSGrowth3yr, RevenueGrowth3yr, NPMGrowth3yr];
          {boolPass, boolCondition}]

In[∘]:= YahooSymbol[symbols_] := Module[{result, newsymbols, exchange, tickers},
          result = StringSplit[symbols, ":"];
          If[Dimensions@symbols === {},
            exchange = First@result;
            tickers = result[[2]];,
            exchange = result[[1, 1]];
            tickers = result[[ ;; , 2]];];
          If[exchange === "NYSE" || exchange === "NASDAQ",
            newsymbols = tickers;,
            If[Dimensions@symbols === {},
              newsymbols = StringJoin[tickers, ".", exchange];,
              newsymbols = StringJoin[#, ".", exchange] & /@ tickers;];];
          newsymbols]

In[∘]:= ConditionTableHeadings = {{Style["Overall Result", Bold],
            "Current price is above 150 day MA and 200 day moving averages",
            "150-day moving average is above the 200-day moving average",
            "200-day moving average line is trending up for at least 1 month",
            "50-day moving average is above both the 150-day and 200-day moving averages",
            "Current stock price is trading above the 50-day moving average",
            "Current stock price is at least 30 percent above its 52-week low",
            "Current stock price is within at least 25 percent of its 52-week high",
            "EPS YOY growth +10% (last 3 years)",
            "Revenue YOY growth +10% (last 3 years)",
            "Net Profit Margin YOY growth +10% (last 3 years)"}, None};

        PrintTestResult :=
          TableForm[Flatten@StockCriteria[ClosePrice, MAFast, MAMedium, MASlow, MASlowLag1M,
              High52Week, Low52Week, EPSGrowth3yr, RevenueGrowth3yr, NPMGrowth3yr],
            TableHeadings -> ConditionTableHeadings];
```