

**tree and workflow for programmer**

Here is a **workflow tree** and **annotated code workflow** designed for your programmer, based on the comprehensive analysis framework and charting scripts included in your attached files. This will clearly show how data moves from ingestion, through analysis, to chart creation and output [[script.py](http://script.py)][script\_1.py][script\_2.py][script\_3.py][script\_4.py][script\_5.py][script\_6.py][script\_7.py][script\_8.py][script\_9.py][script\_10.py][stock\_analysis\_framework.py][chart\_script.py][chart\_script\_1.py][chart\_script\_2.py][chart\_script\_3.py][chart\_script\_4.py].

**Stock Market Analysis & Visualization Workflow Tree**

1. Data Ingestion  
 └─ Load trading metrics and universe data  
 ├─ basic\_calculation\_daily\_YYYYMMDD.csv  
 └─ tradingview\_universe.csv  
  
2. Data Preparation  
 ├─ Merge datasets on ticker/symbol  
 ├─ Parse index membership  
 ├─ Engineer columns (sector, industry, market cap)  
 └─ Derive flags for index membership (SP500, NASDAQ 100, etc.)  
  
3. Analysis & Aggregation  
 ├─ Sector and industry analysis  
 ├─ Index membership performance  
 ├─ Risk/volatility and technical indicators  
 ├─ Multi-period performance (1D, 5D, 1M, 1Y, etc.)  
 ├─ Top/bottom performer selection  
 └─ Prepare summary tables for visualization  
  
4. Output Generation  
 ├─ Save summary tables as .csv files:  
 │ sector\_performance.csv (sector performance)  
 │ sp500\_comparison.csv (S&P500/non-S&P500 comparison)  
 │ technical\_analysis\_by\_sector.csv (technical by sector)  
 │ risk\_return\_analysis.csv (risk/return metrics)  
 │ index\_analysis.csv (major indices performance)  
 │ time\_performance\_analysis.csv (period-by-period stats)  
 │ top\_performers\_1year.csv (top 1-year performers)  
 └─ Export config: analysis\_config.json  
  
5. Visualization (Chart Creation)  
 ├─ chart\_script.py (Sector 1Y Performance Bar Chart)  
 ├─ chart\_script\_1.py (Market Cap vs. Performance Scatter)  
 ├─ chart\_script\_2.py (Index Membership Comparison Bar Chart)  
 ├─ chart\_script\_3.py (Risk vs. Return Scatter Plot)  
 └─ chart\_script\_4.py (Multi-period Performance Line/Error Chart)  
  
6. Integration & Expansion  
 ├─ Use USAGE\_GUIDE.md for framework instructions  
 ├─ Add new trading CSVs as needed  
 ├─ Update config for more metrics/indexes  
 └─ Extend chart scripts for custom visualization

[[script.py](http://script.py)][script\_2.py][script\_3.py][script\_4.py][script\_5.py][script\_8.py][script\_10.py][stock\_analysis\_framework.py][chart\_script.py][chart\_script\_1.py][chart\_script\_2.py][chart\_script\_3.py][chart\_script\_4.py]

**Annotated Code Workflow for Implementation**

**1. Load Data**

* Read latest (or all) trading metric CSVs and universe file using pandas.
* Merge on ticker/Symbol columns for unified dataset.

**2. Prepare and Clean Data**

* Parse indices field (as list).
* Engineer key columns:
  + Market cap (converted to billions)
  + Flags for index membership (SP500, NASDAQ100, etc.)
  + RSI category

**3. Analyze**

* Group by sector/industry and compute performance metrics.
* Compare index member returns to non-members.
* Analyze volatility (ATR), technical indicators, market cap.
* Multi-period statistics (mean, std, positive/negative count for returns).

**4. Save Outputs**

* Write summary tables to .csv for each analysis target.
* Export config file for framework control.

**5. Create Charts (Visualization)**

* Use output .csv tables in chart scripts:
  + Bar chart (sector 1Y return)
  + Scatter plot (market cap vs. performance)
  + Grouped bar (index membership)
  + Bubble/scatter (risk vs. return)
  + Line chart (multi-period mean/std)

**6. Expansion/Customization**

* Add new data files as basic\_calculation\_daily\_YYYYMMDD.csv
* Modify analysis\_config.json to track metrics/columns/indexes
* Use/extend USAGE\_GUIDE.md for instructions and further charting

**Diagram for Code/Data Flow**

Trading CSV + Universe CSV  
 ↓  
 DataFrame (merge)  
 ↓  
 ┌──────────────┬────────┬─────────────────────┐  
 │ Data Prep │ Sector │ Index/Risk/Technical│  
 │ & Cleaning │Analysis│Analysis/Aggregation │  
 └──────────────┴────────┴─────────────────────┘  
 ↓  
 Save .csv Tables  
 ↓  
 Chart Scripts (sector, index, risk-return, period)  
 ↓  
 Visualizations (.png, .svg)  
 ↓  
 Dashboard/Reporting/Custom Analysis

[stock\_analysis\_framework.py][script\_10.py][chart\_script.py][chart\_script\_1.py][chart\_script\_2.py][chart\_script\_3.py][chart\_script\_4.py]

**Key Files & Functions**

|  |  |
| --- | --- |
| File | Purpose/Function |
| stock\_analysis\_framework.py | Main expandable analysis framework |
| analysis\_config.json | Configuration for metrics, columns, indices |
| sector\_performance.csv | Sector-level performance summary |
| index\_analysis.csv | Index membership vs performance data |
| risk\_return\_analysis.csv | Volatility/risk analysis table |
| technical\_analysis\_by\_sector.csv | Sector-level technicals summary |
| time\_performance\_analysis.csv | Multi-period return stats |
| top\_performers\_1year.csv | Top 1Y performing stocks |
| chart\_script\*.py | Individual plotting scripts for main visualizations |

This tree and workflow can be given to any programmer to implement, maintain, or expand your analysis suite and to build additional charts or dashboards quickly and reliably [stock\_analysis\_framework.py][script\_10.py][chart\_script.py][chart\_script\_1.py][chart\_script\_2.py][chart\_script\_3.py][chart\_script\_4.py].

⁂

1. [script.py](http://script.py)

1. script\_9.py

1. script\_10.py

1. USAGE\_GUIDE.md

1. chart\_script.py

1. chart\_script\_1.py

1. chart\_script\_2.py

1. chart\_script\_3.py

1. chart\_script\_4.py

1. stock\_analysis\_framework.py

1. risk\_return\_analysis.csv

1. script\_1.py

1. sector\_performance.csv

1. sp500\_comparison.csv

1. stock\_analysis\_framework.py

1. technical\_analysis\_by\_sector.csv

1. time\_performance\_analysis.csv

1. top\_performers\_1year.csv

1. script\_2.py

1. script\_3.py

1. script\_4.py

1. script\_5.py

1. script\_6.py

1. script\_7.py

1. script\_8.py