Systematic Trading Framework - Code Reference

Generated: 2026-02-06 (local)

Scope

This document explains each non-notebook file in the repository, with a focus on code files. For Python modules, it documents every top-level def and class, plus methods inside classes, and describes where each file sits in the project workflow. Notebooks are excluded by request.

Workflow Overview

Data ingestion and validation -> Feature engineering -> Modeling -> Signal generation -> Backtesting -> Evaluation -> Logging and diagnostics. Utilities and configs support the pipeline.

File: .gitattributes

Workflow role: Repository hygiene and git configuration

Type: text or metadata file.

File: .gitignore

Workflow role: Repository hygiene and git configuration

Type: text or metadata file.

File: README.md

Workflow role: Project overview and usage guide

Type: Markdown documentation

File: config/.gitkeep

Workflow role: Experiment configuration

Type: text or metadata file.

File: config/base/daily.yaml

Workflow role: Experiment configuration

Type: YAML configuration

Top-level keys: backtest, data, logging, risk

File: config/experiments/lgbm_spy.yaml

Workflow role: Experiment configuration

Type: YAML configuration

Top-level keys: backtest, data, extends, features, logging, model, risk, signals

File: config/experiments/trend_spy.yaml

Workflow role: Experiment configuration

Type: YAML configuration

Top-level keys: backtest, data, extends, features, logging, model, risk, signals

File: data/.gitkeep

Workflow role: Repository structure placeholder

Type: text or metadata file.

File: data/metadata/.gitkeep

Workflow role: Repository structure placeholder

Type: text or metadata file.

File: data/processed/.gitkeep

Workflow role: Repository structure placeholder

Type: text or metadata file.

File: data/raw/.gitkeep

Workflow role: Repository structure placeholder

Type: text or metadata file.

File: logs/.gitkeep

Workflow role: Repository structure placeholder

Type: text or metadata file.

File: logs/experiments/trend_spy_v1_20260201_211036/config_used.yaml

Workflow role: Run artifacts (backtest outputs)

Type: YAML configuration

Top-level keys: - step, backtest, config_path, data, features, logging, model, risk, signals

File: logs/experiments/trend_spy_v1_20260201_211036/equity_curve.csv

Workflow role: Run artifacts (backtest outputs)

Type: CSV artifact

Header: Date,equity

File: logs/experiments/trend_spy_v1_20260201_211036/positions.csv

Workflow role: Run artifacts (backtest outputs)

Type: CSV artifact

Header: Date,0

File: logs/experiments/trend_spy_v1_20260201_211036/returns.csv

Workflow role: Run artifacts (backtest outputs)

Type: CSV artifact

Header: Date,0

File: logs/experiments/trend_spy_v1_20260201_211036/summary.json

Workflow role: Run artifacts (backtest outputs)

Type: JSON artifact

Top-level keys: model_meta, summary

File: logs/experiments/trend_spy_v1_20260201_211036/turnover.csv

Workflow role: Run artifacts (backtest outputs)

Type: CSV artifact

Header: Date,0

File: output/pdf/systematic_trading_framework_summary.pdf

Workflow role: Generated artifacts

Type: binary artifact (image or PDF).

Not parsed for code. Included as generated output.

File: plots/equity_curve_comparison.png

Workflow role: Generated plots

Type: binary artifact (image or PDF).

Not parsed for code. Included as generated output.

File: requirements.txt

Workflow role: Python dependencies

Type: text or metadata file.

File: src/__init__.py

Workflow role: Misc

No top-level defs or classes.

File: src/backtesting/__init__.py

Workflow role: Backtesting and performance accounting

Imports:

engine, strategies

No top-level defs or classes.

File: src/backtesting/engine.py

Workflow role: Backtesting and performance accounting

Imports:

__future__, dataclasses, numpy, pandas, src.risk.controls, src.risk.position_sizing, typing

Definitions:

`Class: BacktestResult`

No docstring. Inferred purpose: Utility or helper function.

`Function: _compute_summary(returns, periods_per_year)`

No docstring. Inferred purpose: Utility or helper function.

```
Function: run_backtest(df, signal_col, returns_col, returns_type,
cost_per_unit_turnover, slippage_per_unit_turnover, target_vol, vol_col,
max_leverage, dd_guard, max_drawdown, cooloff_bars, periods_per_year)
```

Simple vectorized backtest with optional vol targeting, slippage, and drawdown guard. Returns are interpreted as simple returns by default. If returns_type="log", they are converted to simple returns via expm1 for PnL accounting.

File: src/backtesting/strategies.py

Workflow role: Backtesting and performance accounting

Imports:

__future__, pandas, src.risk.position_sizing, src.signals

Definitions:

`Function: buy_and_hold_signal(df, signal_name)`

Long-only buy-and-hold signal.

`Function: trend_state_long_only_signal(df, state_col, signal_name)`

Long-only signal based on a trend state column (expects 1 for bull).

`Function: trend_state_signal(df, state_col, signal_name, mode)`

Trend-state strategy wrapper (supports long/short/hold modes).

`Function: rsi_strategy(df, rsi_col, buy_level, sell_level, signal_name, mode)`

RSI strategy wrapper (supports long/short/hold modes).

```
Function: momentum_strategy(df, momentum_col, long_threshold, short_threshold,
signal_name, mode)
```

Momentum strategy wrapper (supports long/short/hold modes).

`Function: stochastic_strategy(df, k_col, buy_level, sell_level, signal_name, mode)`

Stochastic %K strategy wrapper (supports long/short/hold modes).

`Function: volatility_regime_strategy(df, vol_col, quantile, signal_name, mode)`

Volatility regime strategy wrapper (supports long/short/hold modes).

`Function: probabilistic_signal(df, prob_col, signal_name, upper, lower)`

Map probability forecasts to {-1,0,1} signal with dead-zone.

`Function: conviction_sizing_signal(df, prob_col, signal_name, clip)`

Linear map prob∈[0,1] to exposure∈[-clip, clip]: exposure = clip * (prob - 0.5) * 2

`Function: regime_filtered_signal(df, base_signal_col, regime_col, signal_name, active_value)`

Keep base signal only when regime_col == active_value (else 0).

`Function: vol_targeted_signal(df, signal_col, vol_col, target_vol, max_leverage, signal_name)`

Scale signal by volatility targeting using scale_signal_by_vol.

File: src/data/__init__.py

Workflow role: Data ingestion and validation

No top-level defs or classes.

File: src/data/loaders.py

Workflow role: Data ingestion and validation

Imports:

__future__, pandas, src.data.providers.alphavantage, src.data.providers.yahoo, typing

Definitions:

`Function: load_ohlcv(symbol, start, end, interval, source, api_key)`

Parameters ---------- symbol : str Ticker (π.χ. "SPY", "AAPL", "BTC-USD"). start : str | None Start date (π.χ. "2010-01-01"). end : str | None End date (π.χ. "2025-01-01"). interval : str "1d", "1h", "5m", κλπ (■πως τα υποστηρ■ζει το yfinance). source : Literal["yahoo", "alpha"] "yahoo" (default) ■ "alpha" για Alpha Vantage FX. api_key : str | None Απαιτε■ται για source="alpha" (■ env ALPHAVANTAGE_API_KEY).

File: src/data/providers/__init__.py

Workflow role: Data ingestion and validation

Imports:

alphavantage, base, yahoo

No top-level defs or classes.

File: src/data/providers/alphavantage.py

Workflow role: Data ingestion and validation

Imports:

__future__, dataclasses, os, pandas, requests, src.data.providers.base, typing

Definitions:

`Class: AlphaVantageFXProvider`

Lightweight wrapper around Alpha Vantage FX_DAILY endpoint.

`Method: get_ohlcv(self, symbol, start, end, interval)`

No docstring. Inferred purpose: Utility or helper function.

File: src/data/providers/base.py

Workflow role: Data ingestion and validation

Imports:

__future__, abc, pandas

Definitions:

`Class: MarketDataProvider`

No docstring. Inferred purpose: Utility or helper function.

`Method: get_ohlcv(self, symbol, start, end, interval)`

Fetch OHLCV data

File: src/data/providers/yahoo.py

Workflow role: Data ingestion and validation

Imports:

__future__, dataclasses, pandas, src.data.providers.base, yfinance

Definitions:

`Class: YahooFinanceProvider`

No docstring. Inferred purpose: Utility or helper function.

`Method: get_ohlcv(self, symbol, start, end, interval)`

No docstring. Inferred purpose: Utility or helper function.

File: src/data/validation.py

Workflow role: Data ingestion and validation

Imports:

__future__, numpy, pandas, typing

Definitions:

`Function: validate_ohlcv(df, required_columns, allow_missing_volume)`

No docstring. Inferred purpose: Validates input data integrity and raises on violations.

File: src/evaluation/__init__.py

Workflow role: Evaluation and reporting (minimal stub)

No top-level defs or classes.

File: src/experiments/__init__.py

Workflow role: Experiment orchestration and model integration

Imports:

runner

No top-level defs or classes.

File: src/experiments/models.py

Workflow role: Experiment orchestration and model integration

Imports:

__future__, lightgbm, numpy, pandas, src.models.lightgbm_baseline, typing

Definitions:

`Function: infer_feature_columns(df, explicit_cols, exclude)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _build_forward_return_target(df, target_cfg)`

No docstring. Inferred purpose: Utility or helper function.

`Function: train_lightgbm_classifier(df, model_cfg, returns_col)`

No docstring. Inferred purpose: Trains a model or creates a fitted estimator.

File: src/experiments/registry.py

Workflow role: Experiment orchestration and model integration

Imports:

__future__, pandas, src.backtesting.strategies, src.experiments.models, src.features, src.features.technical.indicators, src.features.technical.momentum, src.features.technical.oscillators, src.features.technical.trend, typing

Definitions:

`Function: get_feature_fn(name)`

No docstring. Inferred purpose: Utility or helper function.

`Function: get_signal_fn(name)`

No docstring. Inferred purpose: Creates or transforms a trading signal.

`Function: get_model_fn(name)`

No docstring. Inferred purpose: Utility or helper function.

File: src/experiments/runner.py

Workflow role: Experiment orchestration and model integration

Imports:

__future__, dataclasses, datetime, json, pandas, pathlib, src.backtesting.engine, src.data.loaders, src.data.validation, src.experiments.registry, src.utils.config, src.utils.paths, typing, yaml

Definitions:

`Class: ExperimentResult`

No docstring. Inferred purpose: Utility or helper function.

`Function: _apply_feature_steps(df, steps)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _apply_model_step(df, model_cfg, returns_col)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _apply_signal_step(df, signals_cfg)`

No docstring. Inferred purpose: Creates or transforms a trading signal.

`Function: _resolve_vol_col(df, backtest_cfg, risk_cfg)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _validate_returns_series(returns, returns_type)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _save_artifacts(run_dir, cfg, data, bt, model_meta)`

No docstring. Inferred purpose: Utility or helper function.

`Function: run_experiment(config_path)`

No docstring. Inferred purpose: Runs a pipeline or process end to end.

File: src/features/.gitkeep

Workflow role: Feature engineering

Type: text or metadata file.

File: src/features/__init__.py

Workflow role: Feature engineering

Imports:

lags, returns, technical.trend, volatility

No top-level defs or classes.

File: src/features/lags.py

Workflow role: Feature engineering

Imports:

__future__, pandas, typing

Definitions:

`Function: add_lagged_features(df, cols, lags, prefix)`

Add lagged versions of specified columns.

File: src/features/returns.py

Workflow role: Feature engineering

Imports:

__future__, numpy, pandas

Definitions:

`Function: compute_returns(prices, log, dropna)`

r_t = P_t / P_{t-1} - 1 (log=False) r_t = log(P_t / P_{t-1}) (log=True)

`Function: add_close_returns(df, log, col_name)`

Parameters ---------- df : pd.DataFrame OHLCV dataframe log : bool If True -> log-returns, else returns. col_name : str | None Name of the returns column to add. If None, uses "close_logret" or "close_ret".

File: src/features/technical/__init__.py

Workflow role: Feature engineering

Imports:

indicators, momentum, oscillators, trend

No top-level defs or classes.

File: src/features/technical/indicators.py

Workflow role: Feature engineering

Imports:

__future__, numpy, pandas, typing

Definitions:

`Function: compute_true_range(high, low, close)`

True range as max of (high-low, |high-prev_close|, |low-prev_close|).

`Function: compute_atr(high, low, close, window, method)`

Average True Range (ATR). method: 'wilder' (EWMA) or 'simple' (SMA).

`Function: add_bollinger_bands(close, window, n_std)`

Bollinger bands and derived features: upper, lower, band_width, percent_b.

`Function: compute_macd(close, fast, slow, signal)`

MACD line, signal line, histogram.

`Function: compute_ppo(close, fast, slow, signal)`

Percentage Price Oscillator: normalized MACD.

`Function: compute_roc(close, window)`

Rate of Change: (P_t / P_{t-w}) - 1.

`Function: compute_volume_zscore(volume, window)`

Rolling z-score of volume.

`Function: compute_adx(high, low, close, window)`

ADX with DI+, DI- using Wilder smoothing.

`Function: compute_mfi(high, low, close, volume, window)`

Money Flow Index (uses typical price * volume).

```
Function: add_indicator_features(df, price_col, high_col, low_col, volume_col,
bb_window, bb_nstd, macd_fast, macd_slow, macd_signal, ppo_fast, ppo_slow,
ppo_signal, roc_windows, atr_window, adx_window, vol_z_window, include_mfi)
```

Add a bundle of classic indicators to an OHLCV dataframe.

File: src/features/technical/momentum.py

Workflow role: Feature engineering

Imports:

__future__, numpy, pandas, typing

Definitions:

`Function: compute_price_momentum(prices, window)`

Price momentum: P_t / P_{t-window} - 1

`Function: compute_return_momentum(returns, window)`

Return-based momentum: sum of returns over window. (For log-returns: additive)

`Function: compute_vol_normalized_momentum(returns, volatility, window, eps)`

Volatility-normalized momentum: sum of returns / current volatility

```
Function: add_momentum_features(df, price_col, returns_col, vol_col, windows,
inplace)
```

Προσθ■τει momentum features:

File: src/features/technical/oscillators.py

Workflow role: Feature engineering

Imports:

__future__, numpy, pandas, typing

Definitions:

`Function: compute_rsi(prices, window, method)`

RSI (Relative Strength Index).

`Function: compute_stoch_k(close, high, low, window)`

Stochastic %K:

`Function: compute_stoch_d(k, smooth)`

Stochastic %D: moving average του %K.

`Function: add_oscillator_features(df, price_col, high_col, low_col, rsi_windows, stoch_windows, stoch_smooth, inplace)`

Features: - {price_col}_rsi_{w} - {price_col}_stoch_k_{w} - {price_col}_stoch_k_{w}_d{stoch_smooth}

File: src/features/technical/trend.py

Workflow role: Feature engineering

Imports:

__future__, numpy, pandas, typing

Definitions:

`Function: compute_sma(prices, window, min_periods)`

Simple Moving Average (SMA) .

`Function: compute_ema(prices, span, adjust)`

Exponential Moving Average (EMA) .

`Function: add_trend_features(df, price_col, sma_windows, ema_spans, inplace)`

Προσθ■τει βασικ■ trend features σε OHLCV DataFrame.

`Function: add_trend_regime_features(df, price_col, base_sma_for_sign, short_sma, long_sma, inplace)`

trend "regime" features based on MAs.

File: src/features/volatility.py

Workflow role: Feature engineering

Imports:

__future__, numpy, pandas, typing

Definitions:

`Function: compute_rolling_vol(returns, window, ddof, annualization_factor)`

Rolling realized volatility on a series of returns.

`Function: compute_ewma_vol(returns, span, annualization_factor)`

EWMA volatility (Exponentially Weighted Moving Std)

`Function: add_volatility_features(df, returns_col, rolling_windows, ewma_spans, annualization_factor, inplace)`

Assumes: - df[returns_col]

File: src/models/.gitkeep

Workflow role: Model implementations

Type: text or metadata file.

File: src/models/__init__.py

Workflow role: Model implementations

No top-level defs or classes.

File: src/models/lightgbm_baseline.py

Workflow role: Model implementations

Imports:

__future__, dataclasses, lightgbm, numpy, pandas, src.features.lags, typing

Definitions:

`Function: default_feature_columns(df)`

Select a reasonable feature set if the notebook does not override.

`Class: LGBMBaselineConfig`

No docstring. Inferred purpose: Utility or helper function.

`Function: train_regressor(train_df, feature_cols, target_col, cfg)`

Fit a LightGBM regressor on the provided split.

`Function: predict_returns(model, df, feature_cols, pred_col)`

Generate next-period return predictions and attach to dataframe.

```
Function: prediction_to_signal(df, pred_col, signal_col, long_threshold,
short_threshold)
```

Convert predicted returns to a {-1,0,1} trading signal.

`Function: train_test_split_time(df, train_frac)`

Time-ordered split (no shuffling).

File: src/risk/__init__.py

Workflow role: Risk controls and position sizing

Imports:

controls, position_sizing

No top-level defs or classes.

File: src/risk/controls.py

Workflow role: Risk controls and position sizing

Imports:

__future__, pandas

Definitions:

`Function: compute_drawdown(equity)`

Drawdown series from an equity curve.

`Function: drawdown_cooloff_multiplier(equity, max_drawdown, cooloff_bars, min_exposure)`

When drawdown exceeds max_drawdown, reduce exposure to min_exposure for the next cooloff_bars periods.

File: src/risk/position_sizing.py

Workflow role: Risk controls and position sizing

Imports:

__future__, numpy, pandas, typing

Definitions:

`Function: compute_vol_target_leverage(vol, target_vol, max_leverage, min_leverage, eps)`

Compute leverage to target a given volatility level. leverage = target_vol / vol, clipped to [min_leverage, max_leverage].

`Function: scale_signal_by_vol(signal, vol, target_vol, max_leverage, min_leverage, eps)`

Scale a trading signal by volatility targeting leverage.

File: src/signals/__init__.py

Workflow role: Signal generation and transformation

Imports:

momentum_signal, rsi_signal, stochastic_signal, trend_signal, volatility_signal

No top-level defs or classes.

File: src/signals/momentum_signal.py

Workflow role: Signal generation and transformation

Imports:

__future__, pandas

Definitions:

`Function: compute_momentum_signal(df, momentum_col, long_threshold, short_threshold, signal_col, mode)`

Momentum signal from a precomputed momentum column.

File: src/signals/rsi_signal.py

Workflow role: Signal generation and transformation

Imports:

__future__, pandas

Definitions:

`Function: compute_rsi_signal(df, rsi_col, buy_level, sell_level, signal_col, mode)`

No docstring. Inferred purpose: Computes a derived series or metric from inputs.

File: src/signals/stochastic_signal.py

Workflow role: Signal generation and transformation

Imports:

__future__, pandas

Definitions:

`Function: compute_stochastic_signal(df, k_col, buy_level, sell_level, signal_col, mode)`

Stochastic signal from %K. Long when %K < buy_level, short when %K > sell_level.

File: src/signals/trend_signal.py

Workflow role: Signal generation and transformation

Imports:

__future__, pandas

Definitions:

`Function: compute_trend_state_signal(df, state_col, signal_col, long_value, flat_value, short_value, mode)`

Long-only signal based on a trend state column.

File: src/signals/volatility_signal.py

Workflow role: Signal generation and transformation

Imports:

__future__, pandas

Definitions:

`Function: compute_volatility_regime_signal(df, vol_col, quantile, signal_col, mode)`

Long when volatility is at or below the specified quantile, short when above (if mode allows shorts).

File: src/utils/__init__.py

Workflow role: Utilities and infrastructure

No top-level defs or classes.

File: src/utils/config.py

Workflow role: Utilities and infrastructure

Imports:

__future__, os, pathlib, src.utils.paths, typing, yaml

Definitions:

`Class: ConfigError`

Raised for invalid or inconsistent experiment configs.

`Function: _resolve_config_path(config_path)`

Resolve a config path relative to CONFIG_DIR and verify it exists.

`Function: _load_yaml(path)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _deep_update(base, updates)`

Recursively merge mappings; lists and scalars are overwritten.

`Function: _load_with_extends(path, seen)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _default_risk_block(risk)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _default_backtest_block(backtest)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _resolve_logging_block(logging_cfg, config_path)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _validate_data_block(data)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _inject_api_key_from_env(data)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _validate_features_block(features)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _validate_model_block(model)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _validate_signals_block(signals)`

No docstring. Inferred purpose: Creates or transforms a trading signal.

`Function: _validate_risk_block(risk)`

No docstring. Inferred purpose: Utility or helper function.

`Function: _validate_backtest_block(backtest)`

No docstring. Inferred purpose: Utility or helper function.

`Function: load_experiment_config(config_path)`

Load an experiment YAML, apply inheritance, defaults, validation, and resolve logging paths. Returns a plain dict ready for use.

File: src/utils/paths.py

Workflow role: Utilities and infrastructure

Imports:

__future__, pathlib

Definitions:

`Function: in_project(*parts)`

No docstring. Inferred purpose: Utility or helper function.

`Function: ensure_directories_exist()`

No docstring. Inferred purpose: Utility or helper function.

`Function: describe_paths()`

No docstring. Inferred purpose: Utility or helper function.

File: tests/.gitkeep

Workflow role: Testing and verification

Type: text or metadata file.

File: tests/conftest.py

Workflow role: Testing and verification

Imports:

__future__, pathlib, sys

No top-level defs or classes.

File: tests/test_core.py

Workflow role: Testing and verification

Imports:

numpy, pandas, pytest, src.backtesting.engine, src.data.validation, src.features.returns, src.features.technical.trend

Definitions:

`Function: test_compute_returns_simple_and_log()`

No docstring. Inferred purpose: Utility or helper function.

`Function: test_add_trend_features_columns()`

No docstring. Inferred purpose: Utility or helper function.

`Function: test_validate_ohlcv_flags_invalid_high_low()`

No docstring. Inferred purpose: Utility or helper function.

`Function: test_run_backtest_costs_and_slippage_reduce_returns()`

No docstring. Inferred purpose: Utility or helper function.

`Function: test_run_backtest_log_returns_are_converted()`

No docstring. Inferred purpose: Utility or helper function.