🤖 Trading Bot - Schulprojekt

Ein umfassender Trading Bot mit RSI- und SMA-Strategien für automatisiertes Backtesting mit Yahoo Finance Daten.

Entwickelt von: Silvan
Projekt: ABU Schulprojekt
Datum: September 2025

📋 Inhaltsverzeichnis

🎯 Projektübersicht

Dieser Trading Bot wurde als Schulprojekt entwickelt, um die Grundlagen des algorithmischen Handels und der quantitativen Finanzanalyse zu verstehen. Der Bot implementiert zwei bewährte Trading-Strategien:

- **RSI (Relative Strength Index)**: Momentum-basierte Strategie
- **SMA (Simple Moving Average)**: Trend-folgende Strategie

🎓 Lernziele

- Verstehen von Trading-Algorithmen
- Praktische Anwendung von Python in der Finanzwelt
- Datenanalyse mit Pandas und NumPy
- Backtesting und Performance-Messung
- Clean Code und Dokumentation

✨ Features

🔥 Kern-Features

- **Zwei Trading-Strategien**: RSI und SMA mit konfigurierbaren Parametern
- **Historische Daten**: Automatischer Download von Yahoo Finance
- **Backtesting-Engine**: Realistische Simulation mit Kommissionen und Slippage
- **Performance-Metriken**: Umfassende Analyse der Ergebnisse
- **Logging**: Detaillierte Protokollierung aller Aktivitäten
- **Konfigurierbar**: Einfache Anpassung aller Parameter

📊 Analytics

- **Rendite-Berechnung**: Absolute und prozentuale Gewinne/Verluste
- **Risiko-Metriken**: Drawdown, Volatilität, Sharpe Ratio
- **Trade-Statistiken**: Anzahl Trades, Gewinnrate, etc.
- **Vergleichsanalyse**: Direkter Vergleich der Strategien

🛡️ Robustheit

- **Fehlerbehandlung**: Umfassende Exception-Behandlung
- **Datenvalidierung**: Automatische Prüfung der Marktdaten
- **Logging**: Vollständige Nachverfolgbarkeit
- **Tests**: Unit Tests für alle Komponenten

🏗️ Architektur

```
trading-bot/
├── src/                          # Hauptquellcode
│   ├── main.py                   # Eingangspoint des Programms
│   ├── strategies/               # Trading-Strategien
│   │   ├── __init__.py
│   │   ├── rsi_strategy.py       # RSI-Strategie
│   │   └── sma_strategy.py       # SMA-Strategie
│   ├── data/                     # Datenquellen
│   │   ├── __init__.py
│   │   └── yahoo_finance.py      # Yahoo Finance API
│   ├── backtesting/              # Backtesting-Engine
│   │   ├── __init__.py
│   │   └── engine.py             # Hauptbacktesting-Logik
│   └── utils/                    # Hilfsfunktionen
│       ├── __init__.py
│       └── indicators.py         # Technische Indikatoren
├── tests/                        # Unit Tests
│   ├── __init__.py
│   └── test_strategies.py        # Tests für Strategien
├── config.py                     # Konfigurationsdatei
├── requirements.txt              # Python-Abhängigkeiten
└── README.md                     # Diese Datei
```

🧩 Komponenten-Übersicht

1. **Data Layer**: Yahoo Finance API Integration
2. **Strategy Layer**: RSI und SMA Implementierungen
3. **Backtesting Engine**: Portfolio-Management und Trade-Simulation
4. **Utils**: Technische Indikatoren und Hilfsfunktionen
5. **Configuration**: Zentrale Parameter-Verwaltung

🚀 Installation

Voraussetzungen

- **Python 3.8+** (empfohlen: Python 3.11)
- **pip** (Python Package Manager)
- **Internetverbindung** (für Yahoo Finance Daten)

Schritt-für-Schritt Installation

1. **Repository klonen/herunterladen**

```
# Falls du Git verwendest:
git clone [repository-url]
cd trading-bot

# Oder einfach die Dateien herunterladen und entpacken
```

2. **Virtuelle Umgebung erstellen** (empfohlen)

```
# Windows
python -m venv trading_env
trading_env\Scripts\activate

# macOS/Linux
python3 -m venv trading_env
source trading_env/bin/activate
```

3. **Abhängigkeiten installieren**

```
pip install -r requirements.txt
```

4. **Installation verifizieren**

```
python -c "import yfinance, pandas, numpy; print('✅ Alle Module
erfolgreich installiert!')"
```

🍎 macOS Spezifische Installation

```
# Falls du Homebrew hast:
brew install python3
pip3 install -r requirements.txt

# Für M1/M2 Macs (ARM):
arch -arm64 pip install -r requirements.txt
```

📖 Verwendung

🚀 Schnellstart

1. **Bot starten**

```
cd trading-bot
python src/main.py
```

2. **Erwartete Ausgabe**

```
==========================================
TRADING BOT BACKTESTING ERGEBNISSE
==========================================

Symbol: AAPL
Zeitraum: 2024-09-16 bis 2025-09-16
Anfangskapital: $10,000.00

📊 RSI STRATEGIE ERGEBNISSE:
   Endkapital: $12,450.32
   Gesamtrendite: 24.50%
   Anzahl Trades: 15
   Gewinnende Trades: 9
   Max Drawdown: 8.23%

📈 SMA STRATEGIE ERGEBNISSE:
   Endkapital: $11,230.15
   Gesamtrendite: 12.30%
   Anzahl Trades: 8
   Gewinnende Trades: 5
   Max Drawdown: 12.45%

🏆 BESTE STRATEGIE: RSI mit 24.50% Rendite
==========================================
```

⚙️ Konfiguration anpassen

1. **Symbol ändern** (in `config.py`):

```
SYMBOL = "TSLA"  # Tesla statt Apple
```

2. **Parameter anpassen**:

```
# RSI Strategie
RSI_PERIOD = 21          # Längere Periode
RSI_OVERSOLD = 25        # Aggressiverer Einstieg
RSI_OVERBOUGHT = 75      # Konservativerer Ausstieg

# SMA Strategie
SMA_SHORT_WINDOW = 5     # Kürzeres Fenster
SMA_LONG_WINDOW = 20     # Längeres Fenster
```

📊 Strategien

📈 RSI (Relative Strength Index) Strategie

Der RSI ist ein Momentum-Oszillator, der zwischen 0 und 100 schwankt.

📋 Trading-Regeln

- **Kaufsignal**: RSI < 30 (überverkauft)
- **Verkaufssignal**: RSI > 70 (überkauft)
- **Periode**: 14 Tage (konfigurierbar)

📊 SMA (Simple Moving Average) Strategie

Die SMA-Strategie basiert auf zwei gleitenden Durchschnitten.

📋 Trading-Regeln

- **Kaufsignal**: Kurzer SMA > Langer SMA (Golden Cross)
- **Verkaufssignal**: Kurzer SMA < Langer SMA (Death Cross)
- **Standard**: 10-Tage und 30-Tage SMA

🧪 Testing

🔬 Unit Tests ausführen

```
# Alle Tests ausführen
python -m pytest tests/ -v

# Spezifische Tests
```

```
python -m pytest tests/test_strategies.py -v

# Mit Coverage-Report
python -m pytest tests/ --cov=src --cov-report=html
```

⚙️ Konfiguration

📋 Haupt-Konfigurationsdatei: `config.py`

```
class Config:
    # Trading Parameter
    SYMBOL = "AAPL"                    # Handelssymbol
    INITIAL_CAPITAL = 10000.0          # Startkapital
    COMMISSION = 0.001                 # 0.1% Kommission
    SLIPPAGE = 0.0005                  # 0.05% Slippage

    # RSI Parameter
    RSI_PERIOD = 14                    # RSI Berechnungsperiode
    RSI_OVERSOLD = 30                  # Überverkauft-Schwelle
    RSI_OVERBOUGHT = 70                # Überkauft-Schwelle

    # SMA Parameter
    SMA_SHORT_WINDOW = 10              # Kurzer MA
    SMA_LONG_WINDOW = 30               # Langer MA

    # Backtesting Parameter
    BACKTEST_DAYS = 365                # Tage für Backtesting
    DATA_INTERVAL = "1d"               # Datenintervall
```

🔧 Entwicklung

🏗️ Entwicklungsumgebung Setup

1. **Code-Qualität Tools**

```
# Code formatieren
black src/ tests/

# Code-Style prüfen
flake8 src/ tests/

# Imports sortieren
isort src/ tests/
```

📚 Lernressourcen

📖 Trading & Finanzwissen

📊 Technische Analyse

- **RSI**: Investopedia RSI Guide
- **Moving Averages**: MA Trading Strategies

🐍 Python & Programmierung

- **Pandas**: Pandas Documentation
- **NumPy**: NumPy User Guide

🤝 Mitwirken

🐛 Bug Reports

Wenn du einen Bug findest:

1. **Issue erstellen** mit detaillierter Beschreibung
2. **Reproduktionsschritte** angeben
3. **Error-Logs** anhängen

📄 Lizenz

Dieses Projekt wurde für Bildungszwecke erstellt und steht unter einer MIT-Lizenz.

Wichtige Hinweise:

- ⚠️ **Nur für Bildungszwecke** - Nicht für echtes Trading verwenden
- 📚 **Kein Finanzberatung** - Immer eigene Recherche durchführen
- 🎓 **Lernprojekt** - Kontinuierliche Verbesserung erwünscht

🚀 Nächste Schritte

Nach dem ersten erfolgreichen Durchlauf:

1. 📊 **Verschiedene Symbole testen** (TSLA, MSFT, SPY)
2. ⚙️ **Parameter optimieren** (RSI-Perioden, SMA-Fenster)
3. 📈 **Neue Strategien hinzufügen** (MACD, Bollinger Bands)
4. 🎨 **Visualisierungen erstellen** (Charts, Performance-Plots)

Happy Trading! 📈 🚀

Entwickelt mit ❤️ für das ABU-Schulprojekt von Silvan