

Automated Binance Futures Trading Bot

- Satvik V

1. Introduction

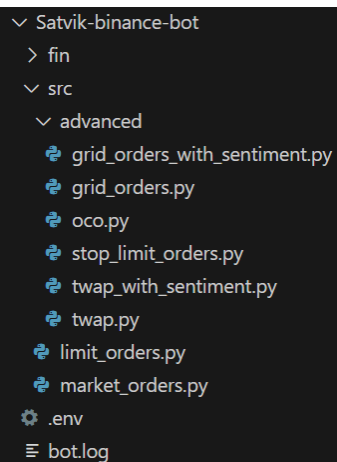
This project implements an automated trading system for Binance Futures Testnet, developed using Python and the Binance Connector SDK.

The goal is to create a modular, strategy-driven trading bot capable of executing and managing different types of orders — such as Market, Limit, Stop-Limit, OCO (One Cancels the Other), TWAP (Time-Weighted Average Price), and Grid strategies — with full validation and optional market sentiment adaptation using the Fear & Greed Index API.

Each module is independent, tested on the Binance Testnet, and follows a robust structure including:

- Environment variable management via `.env`
- Real-time validation for symbol, notional value, and price logic
- Logging to a centralized `bot.log` file at the project root
- Intelligent sentiment-driven strategy adjustments

Project structure:



```
▼ Satvik-binance-bot
  > fin
  ▼ src
    ▼ advanced
      🔗 grid_orders_with_sentiment.py
      🔗 grid_orders.py
      🔗 oco.py
      🔗 stop_limit_orders.py
      🔗 twap_with_sentiment.py
      🔗 twap.py
      🔗 limit_orders.py
      🔗 market_orders.py
    ⚙️ .env
    ≡ bot.log
```

2. Environment Setup

All API keys are securely stored inside `.env`:

- `API_KEY=your_api_key_here`
- `API_SECRET=your_secret_key_here`
- `BASE_URL=https://testnet.binancefuture.com`

This ensures keys are not exposed in the source code and makes the bot deployable across environments.

3. Market Orders Module (market_orders.py)

The market_orders.py module is responsible for placing instant buy or sell orders on Binance Futures Testnet at the current market price.

The script first validates user inputs such as the trading symbol, order side (BUY/SELL), and quantity. It also checks the minimum notional value (100 USDT) before executing the trade. Once validated, it places the order using Binance's REST API and records the outcome in the bot.log file.

To run:- python market_orders.py <symbol> <BUY/SELL> <quantity>

Example Execution:

```
python market_orders.py BTCUSDT BUY 0.002
```

```
PS D:\Satvik\open source projects\Satvik-binance-bot\src> python market_orders.py BTCUSDT BUY 0.002
✅ Market BUY order placed for 0.002 BTCUSDT.
PS D:\Satvik\open source projects\Satvik-binance-bot\src> █
```

4. Limit Orders

The limit_orders.py module is used to place limit buy or sell orders on Binance Futures Testnet. Unlike market orders, a limit order executes only when the market price reaches a specified limit price set by the user.

The script validates the symbol, order side, quantity, and limit price. It also checks whether the total notional value meets Binance's minimum requirement. After validation, it sends the limit order request to Binance and logs the order details in the bot.log file.

To run:- python limit_orders.py <symbol> <BUY/SELL> <quantity> <price>

Example Execution:

```
python limit_orders.py BTCUSDT BUY 0.002 108000
```

```
PS D:\Satvik\open source projects\Satvik-binance-bot\src> python limit_orders.py BTCUSDT BUY 0.002 108000
✅ Limit BUY order placed for 0.002 BTCUSDT at 108000.0.
PS D:\Satvik\open source projects\Satvik-binance-bot\src> █
```

Validation error:

```
PS D:\Satvik\open source projects\Satvik-binance-bot\src> python limit_orders.py BTCUSDT BUY 0.002 10600
❌ Order notional (21.20) is below the minimum required (100.00 USDT).
PS D:\Satvik\open source projects\Satvik-binance-bot\src> █
```

5. Stop-Limit Orders

The stop_limit_orders.py module is used to place conditional orders that execute only after the market reaches a specified stop price. Once the stop price is triggered, a limit order is placed at the defined limit price. This type of order is commonly used to control risk and automate entry or exit points.

The script performs several validations before execution. It ensures the trading symbol exists, checks that all prices and quantities are positive, and verifies that the order notional value meets Binance's minimum requirement. Additionally, it validates the logical relationship between stop and limit prices based on the order side:

- For a **BUY** order, the stop price must be **above** the current market price, and the limit price must be **equal to or greater than** the stop price.
- For a **SELL** order, the stop price must be **below** the current market price, and the limit price must be **equal to or less than** the stop price.

After validation, the script places the stop-limit order and records the details in the bot.log file.

To run:- `python stop_limit_orders.py <symbol> <BUY/SELL> <quantity> <stopPrice> <limitPrice>`

Example Execution:

```
python stop_limit_orders.py BTCUSDT SELL 0.002 109000 108800
```

```
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced> python stop_limit_orders.py BTCUSDT SELL 0.002 109000 108800
✅ Stop-Limit SELL order placed for 0.002 BTCUSDT (Stop: 109000.0, Limit: 108800.0).
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced>
```

Validation error:-

```
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced> python stop_limit_orders.py BTCUSDT BUY two 109000 108800
📁 Logging to: D:\Satvik\open source projects\Satvik-binance-bot\bot.log
❌ Invalid input. Quantity, stop price, and limit price must be numbers.
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced>
```

6. OCO (One-Cancels-the-Other) Orders

The oco_orders.py module is used to place a pair of conditional orders — a **take-profit limit order** and a **stop-loss order** — simultaneously. When one of the two orders executes, the other is automatically canceled. This type of strategy helps manage both profit targets and loss limits in a single command.

The script validates all user inputs, including symbol, side (BUY/SELL), quantity, and both price levels. It also checks the notional value and ensures the correct logical relationship between prices:

- For a **BUY** OCO, the take-profit price must be **below** the current price, and the stop-loss price must be **above** it.
- For a **SELL** OCO, the take-profit price must be **above** the current price, and the stop-loss price must be **below** it.

The module fetches the current market price for reference and shows a quick guide to help the user choose correct input levels. All order details and any errors are logged in the bot.log file at the project root.

To run :- `python oco.py <symbol> <BUY/SELL> <quantity> <takeProfitPrice> <stopLossPrice>`

Example Executions:

```
python oco.py BTCUSDT SELL 0.002 112000 108000
```

```
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced> python oco.py BTCUSDT SELL 0.002 112000 108000
Logging to: D:\Satvik\open source projects\Satvik-binance-bot\bot.log

Current BTCUSDT Price: 109822.90 USDT
For SELL OCO: takeProfit > 109822.90 > stopLoss

OCO SELL order placed for 0.002 BTCUSDT (Take Profit: 112000.0, Stop Loss: 108000.0).
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced> █
```

7. TWAP (Time-Weighted Average Price) Orders

The `twap_orders.py` module automates **Time-Weighted Average Price (TWAP)** trading — a strategy that breaks a large order into smaller chunks and executes them gradually over time to minimize market impact.

The script divides the total quantity specified by the user into smaller equal portions (called **slices**) and places each as a **market order** at fixed time intervals. This helps achieve an average entry or exit price that's close to the market's time-weighted average.

Before execution, the script performs several validations:

- Ensures that the trading symbol exists on Binance.
- Confirms that side (BUY/SELL), quantity, slices, and interval are all positive.
- Checks that each slice meets Binance's **minimum notional requirement** (e.g., at least 100 USDT per order).
- Displays the **current market price**, total quantity, number of slices, and per-slice quantity for confirmation.

Each trade's result (success or failure) is printed to the console and logged in the central `bot.log` file. The program waits the specified interval between consecutive trades and automatically stops once all slices are completed.

Syntax:- `python twap.py <symbol> <BUY/SELL> <total_qty> <num_slices> <interval_seconds>`

Example Execution:-

```
python twap_orders.py BTCUSDT BUY 0.01 5 30
```

```

PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced> python twap.py BTCUSDT BUY 0.01 5 30

📊 Current BTCUSDT Price: 109946.50 USDT
✅ Validation Passed!
→ Total Qty: 0.01, Slices: 5, Chunk Size: 0.002000, Interval: 30s

🚀 Starting TWAP Execution for BTCUSDT
Side: BUY
Total Quantity: 0.01
Split: 5 × 0.002000
Interval: 30 seconds
-----
✅ [1/5] BUY 0.002000 BTCUSDT at ~109946.50 USDT
⌚ Waiting 30s before next order...
✅ [2/5] BUY 0.002000 BTCUSDT at ~109984.30 USDT
⌚ Waiting 30s before next order...
✅ [3/5] BUY 0.002000 BTCUSDT at ~109984.30 USDT
⌚ Waiting 30s before next order...
✅ [4/5] BUY 0.002000 BTCUSDT at ~109950.00 USDT
⌚ Waiting 30s before next order...
✅ [5/5] BUY 0.002000 BTCUSDT at ~109874.70 USDT

🎉 TWAP Execution Completed Successfully!
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced>

```

8. TWAP (Time-Weighted Average Price) Orders — with Live Sentiment Integration

The `twap_orders_live_sentiment.py` module enhances the standard TWAP (Time-Weighted Average Price) strategy by integrating **real-time market sentiment** using the **Fear & Greed Index API** from *alternative.me*. This allows the trading bot to automatically adjust its trading behavior depending on overall market psychology — becoming more aggressive during “fear” periods and more conservative during “greed.”

The script begins by validating all input parameters, such as:

- Trading symbol, order side (BUY/SELL), total quantity, number of slices, and time interval.
- Ensures each trade slice satisfies Binance’s minimum **notional value** (≥ 100 USDT).
- Displays the **current market price** and splits the total order quantity into equal-sized chunks.

It then fetches the **Fear & Greed Index** (ranging from 0 to 100):

- ≤ 25 (**Extreme Fear**) → the market is undervalued, so the bot increases buy volume (or reduces sell size).
- ≥ 75 (**Extreme Greed**) → the market may be overbought, so the bot reduces buy size (or increases sell size).
- Values between 25 and 75 trigger standard TWAP behavior.

After adjusting the per-slice order size, the bot executes a sequence of **market orders** with a delay between each (defined by `interval_seconds`). Every order’s details, success, or failure are logged to the central `bot.log` file, and a summary is printed once all slices are completed.

Syntax: `python twap_with_sentiment.py <symbol> <BUY/SELL> <total_qty> <num_slices> <interval_seconds>`

Example Execution:-

python twap_with_sentiment.py BTCUSDT BUY 0.01 5 30

```
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced> python twap_with_sentiment.py BTCUSDT BUY 0.01 5 30

📊 Current BTCUSDT Price: 109845.40 USDT
✅ Validation Passed!
➔ Total Qty: 0.01, Slices: 5, Chunk Size: 0.002000, Interval: 30s

📊 Live Fear & Greed Index: 27 (Fear)

🚀 Starting TWAP Execution for BTCUSDT
Market Sentiment: Fear (27)
Side: BUY
Total Quantity: 0.01
Adjusted Chunk: 0.002000
Split: 5 x 0.002000
Interval: 30 seconds
-----
✅ [1/5] BUY 0.002000 BTCUSDT at ~109877.00 USDT
⌚ Waiting 30s before next order...
✅ [2/5] BUY 0.002000 BTCUSDT at ~109941.60 USDT
⌚ Waiting 30s before next order...
✅ [3/5] BUY 0.002000 BTCUSDT at ~109930.20 USDT
⌚ Waiting 30s before next order...
✅ [4/5] BUY 0.002000 BTCUSDT at ~109930.20 USDT
⌚ Waiting 30s before next order...
✅ [5/5] BUY 0.002000 BTCUSDT at ~109968.00 USDT

🎉 TWAP Execution Completed Successfully!
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced>
```

9. Grid Trading Strategy

The grid_orders.py module implements a **Grid Trading Bot**, which is a classic algorithmic trading strategy designed to profit from **price oscillations** within a defined range. It automatically places multiple **BUY** and **SELL** limit orders at evenly spaced price levels, creating a "grid" across the chosen price range.

The idea is simple:

- **BUY** when the market dips to a lower level.
 - **SELL** when it rises to a higher level.
- This allows traders to profit from sideways or mildly volatile markets without predicting exact price direction.

Key Features & Validations:

1. **Symbol Validation:**
Ensures the trading pair (e.g., BTCUSDT) exists on Binance Futures.
2. **Price Range Validation:**
Confirms that the **lower price < upper price** and both are positive.
3. **Grid Count & Quantity Check:**
Requires at least **2 grids** (to form a range) and validates that trade quantity is greater than zero.

4. **Minimum Notional Value:**

Verifies that each order meets Binance's **minimum order value** (typically 100 USDT).

5. **Grid Calculation:**

Divides the range into equal price gaps and creates corresponding grid levels.

6. **Automated Order Placement:**

- Places **BUY** limit orders below the midpoint of the range.
- Places **SELL** limit orders above the midpoint.

7. **Logging & Tracking:**

Each placed order and any encountered error are logged in the central bot.log file for monitoring and debugging.

To use- `python grid_orders.py <symbol> <lower_price> <upper_price> <num_grids> <quantity>`

Ex:-

`python grid_orders.py BTCUSDT 105000 115000 6 0.001`

```
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced> python grid_orders.py BTCUSDT 105000 115000 6 0.001

📊 Current BTCUSDT Price: 109766.20 USDT
✅ Validation Passed!
➔ Range: 105000.0 - 115000.0, Grids: 6, Quantity: 0.001

🚀 Starting Grid Trading Strategy for BTCUSDT
Range: 105000.0 → 115000.0 | Grids: 6 | Qty: 0.001
-----
✅ BUY Limit [1] at 105000.0 for 0.001 BTCUSDT
✅ BUY Limit [2] at 107000.0 for 0.001 BTCUSDT
✅ BUY Limit [3] at 109000.0 for 0.001 BTCUSDT
✅ SELL Limit [1] at 113000.0 for 0.001 BTCUSDT
✅ SELL Limit [2] at 115000.0 for 0.001 BTCUSDT

🎯 Grid Orders Successfully Placed!
💡 The bot will automatically profit from market oscillations.
PS D:\Satvik\open source projects\Satvik-binance-bot\src\advanced>
```

10. bot.log — Centralized Trade & Event Logger

The **bot.log** file serves as the **central audit trail** for all trading actions, system validations, and API communications handled by the Binance Futures Bot.

Every script (Market, Limit, Stop-Limit, OCO, TWAP, and Grid strategies) writes structured logs into this single file, enabling full traceability and performance analysis of trades.

Purpose of bot.log:

- To **record every trading event** (e.g., order placement, validation, or failure).
- To **track errors** such as invalid prices, network issues, or Binance API rejections.
- To **maintain transparency** and allow post-trade analysis or debugging.
- To **monitor sentiment-based strategies**, showing how market mood (Fear/Greed) affects bot decisions.

bot.log

Satvik-binance-bot > bot.log

```
1 2025-10-23 11:28:01,778 - INFO - \u2705 Market BUY order placed for 0.001 BTCUSDT.
2 2025-10-23 11:28:01,782 - INFO - Order response: {'orderId': 6697221939, 'symbol': 'BTCUSDT', 'status': 'NEW',
3 2025-10-23 11:39:13,215 - ERROR - \u274c Failed to place limit order: (400, -4164, "Order's notional must be no
4 2025-10-23 11:39:59,396 - INFO - \u2705 Limit BUY order placed for 0.002 BTCUSDT at 64000.0.
5 2025-10-23 11:39:59,396 - INFO - Order response: {'orderId': 6698630328, 'symbol': 'BTCUSDT', 'status': 'NEW',
6 2025-10-23 11:52:49,011 - ERROR - \u274c Failed to place limit order: (400, -4024, "Limit price can't be lower
7 2025-10-23 12:47:54,056 - INFO - \u2705 Limit SELL order placed for 0.002 BTCUSDT at 109900.0.
8 2025-10-23 12:47:54,061 - INFO - Order response: {'orderId': 6706740577, 'symbol': 'BTCUSDT', 'status': 'NEW',
9 2025-10-23 12:53:33,008 - INFO - \u2705 Market BUY order placed for 0.002 BTCUSDT.
10 2025-10-23 12:53:33,008 - INFO - Order response: {'orderId': 6707396618, 'symbol': 'BTCUSDT', 'status': 'NEW',
11 2025-10-23 12:59:17,690 - INFO - \u2705 Limit SELL order placed for 0.002 BTCUSDT at 109900.0.
12 2025-10-23 12:59:17,690 - INFO - Order response: {'orderId': 6708040459, 'symbol': 'BTCUSDT', 'status': 'NEW',
13 2025-10-23 13:05:15,991 - INFO - \u2705 Stop-Limit SELL order placed for 0.002 BTCUSDT (Stop: 107000.0, Limit:
14 2025-10-23 13:05:15,991 - INFO - Order response: {'orderId': 6708731352, 'symbol': 'BTCUSDT', 'status': 'NEW',
15 2025-10-23 13:09:02,384 - INFO - \u2705 OCO SELL order placed for 0.002 BTCUSDT (Take Profit: 110000.0, Stop Lo
16 2025-10-23 13:09:02,385 - INFO - TP Order: {'orderId': 6709166101, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clie
17 2025-10-23 13:09:02,385 - INFO - SL Order: {'orderId': 6709166350, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clie
18 2025-10-23 13:10:16,038 - INFO - \u2705 OCO BUY order placed for 0.002 BTCUSDT (Take Profit: 104000.0, Stop Lo
19 2025-10-23 13:10:16,038 - INFO - TP Order: {'orderId': 6709308291, 'symbol': 'BTCUSDT', 'status': 'NEW', 'clie
```