**Python Project Directory Structure**

Below is the folder structure for the AI trading bot, designed for PythonAnywhere compatibility. All Python modules are stored in a dedicated directory, and the historical analysis JSON file resides in the user's home directory.

**A computer screen shot of text

AI-generated content may be incorrect.**

* **Environment Variables**: ALPACA\_API\_KEY, ALPACA\_API\_SECRET, and XAI\_API\_KEY are set in .env
* **Log File**: Logs are written to /home/gmafanasiev/ai\_trading\_bot/trade\_alpaca\_grok4.log.
* **JSON File**: Historical data is read from  /home/gmafanasiev/historical\_analysis\_APPL.json

Start (main.py: Init Clients/Logging)

|

V

Load Config (config.py: Modes/Keys/Params)

|

V

Setup Logging (logging\_utils.py: Configure File/Stream Handlers)

|

V

Trading Loop (trading\_loop.py: Session Check/Cooldown)

|

V

Fetch Bars (data\_utils.py: Load JSON Supports/Resistances + Sim or Real Data)

|

V

Get Prediction (prediction.py: Grok-3 API Call with JSON Stats)

|

V

If Signal: Execute Order (order\_execution.py: Submit/Poll/Update Position)

|

V

Analyze Trades (trade\_analysis.py: P/L/Win Rate/Plot Equity)

|

V

Sleep/Repeat or End Session (Close Positions if Needed)

**Module Function**

1. config.py: Stores all parameters (e.g., TRADE\_MODE, thresholds, API keys, position tracking). Central for easy tweaks.
2. main.py: Entry point—sets up logging, inits Alpaca clients, runs trading loop.
3. logging\_utils.py: Configures file/stream handlers with EDT formatter for consistent logs.
4. trading\_loop.py: Core loop—handles session duration, processes symbols (simulated or real-time bars), cooldowns, SL/TP checks, and end-of-session closes.
5. data\_utils.py: Fetches/simulates bar data, prepares input stats for predictions (closes, volumes, returns, etc.).
6. prediction.py: Calls Grok-3 API for price direction prediction (0-1 score) based on stats, adjusts thresholds/risk.
7. order\_execution.py: Submits/simulates orders, polls fills in live mode, updates positions/SL/TP, calculates P/L.
8. trade\_analysis.py: Parses logs for trades/P/L/win rate, generates equity curve plot with timestamps/trade types.

**A diagram of a network

AI-generated content may be incorrect.**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Module Name | Description | Function in Project |
| 1 | config.py | Central configuration file for API keys, trading parameters, and position tracking. | Defines constants, credentials, and global settings like thresholds, risk levels, and API endpoints for all modules to use consistently. |
| 2 | data\_utils.py | Handles fetching and preparing historical and mock bar data for predictions. | Provides OHLCV data from Alpaca or mock sources, calculates stats like ATR and volatility, enabling input for Grok predictions during live or off-hours testing. |
| 3 | prediction.py | Fetches predictions and adjustments from Grok-3, including sentiment analysis. | Generates price direction confidence scores (0-1) using xAI API, integrating sentiment and stats to signal buys/sells in the trading loop. |
| 4 | trade\_analysis.py | Analyzes trade logs for P/L, win rate, equity curve, and drawdown. | Processes log data to compute performance metrics and generate equity plots, used for risk checks and post-session summaries. |
| 5 | order\_execution.py | Manages position retrieval, order placement, and updates with stop-loss/take-profit. | Executes buy/sell orders via Alpaca in paper/live modes, updates in-memory positions, and handles polling for fills, ensuring safe trade execution. |
| 6 | trading\_loop.py | Core loop for real-time trading logic, data fetching, predictions, and order triggers. | Orchestrates the bot's main cycle: fetches data, gets predictions, checks risks, executes trades, and manages sessions/WebSocket for continuous operation. |
| 7 | main.py | Entry point for the bot, setting up clients and running the trading loop. | Initializes Alpaca clients, logging, and signal handling, then starts the trading loop, serving as the bot's launcher. |
| 8 | logging\_utils.py | Configures logging with file and console handlers in EDT time zone. | Sets up consistent logging for the entire project, ensuring debug/info/error messages are timestamped and saved for analysis. |