**Imports Block**

This section imports necessary libraries for asynchronous operations (asyncio), data handling (pandas, numpy, sklearn), API requests (requests), logging, datetime utilities, and Alpaca-specific modules for trading and data access. It sets up the foundation for fetching market data, executing trades, and processing predictions. The AI intent here is to enable data preparation for Grok 3 predictions and Alpaca API interactions for automated trading on AMD.

**Logging Configuration Block**

Configures a logger with INFO level, adding file (trade\_alpaca\_grok4.log) and console handlers for formatted output (%(asctime)s - %(message)s). The file handler flushes for immediate writes. This tracks script execution, errors, and trade details, crucial for debugging AI-assisted decisions and trade outcomes.

**Alpaca Clients Initialization Block**

Defines API credentials (ALPACA\_API\_KEY, ALPACA\_API\_SECRET) for paper trading and initializes TradingClient (for orders/positions) and StockHistoricalDataClient (for data). Logs connection success or errors. The AI intent is to connect to Alpaca for real-time/historical data fetching and order placement, enabling paper trading simulations based on Grok predictions without real money risk.

**Trading Parameters Block**

Sets constants like symbols (AMD), sequence length (60 days for data stats), risk management (1% risk per trade, 5% max position, $1000 cap), stop-loss (2%), take-profit (4%), trailing (2%), thresholds (0.65 buy, 0.35 sell), and adjustment interval (15 seconds). This defines the strategy's risk profile and decision thresholds, where Grok's AI predictions (0.0-1.0) trigger buys/sells if above/below thresholds, aiming for AI-assisted directional trading.

**Global Variables Block**

Initializes open\_positions dict for tracking AMD positions (type, qty, entry price), and sl\_prices/tp\_prices for stop-loss/take-profit. These maintain state for position management, allowing the AI strategy to check and update based on Grok predictions and market data.

**xAI API Credentials Block**

Fetches XAI\_API\_KEY from environment variables, raising an error if missing. This is for authenticating requests to Grok 3's API, central to the AI intent: sending prepared data stats for price direction predictions.

**Fetch Bar Data Function**

An async function to retrieve historical daily bars for a symbol (AMD) using Alpaca's API, with a try-except for errors (e.g., SIP restrictions). Returns a DataFrame with 'close' prices or empty on failure. The AI intent is to gather 60 days of data for stats preparation, feeding into Grok for predictions; errors log and could trigger fallbacks (not implemented here).

**Prepare Grok Input Function**

Processes the DataFrame to scale the last 60 close prices (0-1 range), computes stats (mean, std, min, max, last 5), and returns a dict. Returns None if data is empty. This prepares concise input for Grok 3, focusing the AI on recent trends for price direction prediction.

**Get Grok Prediction Function**

Constructs a prompt with data stats and trade summary, sends to Grok 3 API with retries/backoff. Parses JSON response for prediction (0.0-1.0), threshold (0.45-0.7), and risk (0.005-0.015), defaulting on errors. The core AI intent: Grok analyzes stats and performance to predict AMD direction and suggest adjustments, driving buy/sell decisions.

**Analyze Trades Function**

Parses the log file for today's trades using regex (buy/sell patterns), matches opens/closes to calculate realized P/L (long: sell - buy; short: entry - buy), adds unrealized P/L from current prices, computes total P/L, trades count, and win rate (positive P/L trades ratio). Returns summary dict. Supports AI feedback loop by analyzing past trades for Grok's prompt, improving future predictions.

**Get Position Function**

Queries Alpaca for current position qty for a symbol, returns 0 if none or on error. This checks open AMD positions, essential for the AI strategy to avoid over-trading or decide closes based on Grok predictions.

**Place Order Function**

Places market orders via Alpaca (buy/sell), updates global positions/entry prices for averaging, sets SL/TP, logs details. Handles long/short opens/closes, resets on zero position. The AI intent is executed here: based on Grok predictions exceeding thresholds, it places risk-managed orders (qty from risk amount / SL distance) for AMD trading.

**Trading Logic Function**

An async loop simulating 10 minutes (adjusts every 15 seconds): fetches equity, analyzes trades, gets data/predictions for AMD, updates trailing SL/TP, places/closes orders per thresholds, closes positions at end. Logs everything. The overall AI-assisted trading intent: continuously monitor data, use Grok for predictions, apply risk rules to trade AMD directionally in a paper environment.

**Main Function**

Logs start message and runs trading\_logic async. Entry point to initiate the AI trading loop, focusing on automated, prediction-driven execution for AMD.