

A BEGINNER'S MANUAL

GROK EDITION

Autonomous AI Trading **for** **Beginners**

Build, test, and supervise a hands-off trading bot, step by step, with paper money first

Autonomous AI Trading for Beginners

Build, test, and supervise a hands-off trading bot, step by step, with paper money first

GROK EDITION

Concise edition, revised and fact-checked by Grok Bot in GrokAllManualsV1.

This manual takes you from "what is a stock order?" to a trading bot that runs on a schedule, makes its own decisions, and places orders through a broker's API. It explains every step in plain English, assuming you have never written code or traded before.

It is also deliberately **honest**. Most automated strategies lose money once you count costs, and a bot that runs "100% autonomously" can lose money faster than you can notice. That's why everything in this book starts in **paper trading** (simulated money), adds hard safety limits, and treats going live as an optional final step that most readers should delay or skip.

HOW TO READ THIS MANUAL

Tip — a shortcut or good habit.

Warning — easy to get wrong.

`$ command` — text you type, exactly as shown.

Note — useful background.

Danger — can lose data, money, or access.

`Settings > Privacy` — a menu path to click.

New words appear in **bold** the first time and are collected in the Glossary at the back.

This manual is education, not financial, investment, legal, or tax advice. Trading involves substantial risk of loss, including the loss of all money you put in. Past or simulated performance does not predict future results. Nothing here recommends any security, strategy, broker, or service. Check current rules, fees, and tax treatment with official sources and a qualified professional before you risk real money.

Contents

01 Read This First: The Honest Truth	5
1.1 What "autonomous" means here	5
1.2 Why most bots lose money	6
1.3 The rules this manual follows	6
1.4 Self-check	7
02 Markets in Plain English	8
2.1 What you can trade	8
2.2 How a trade happens	9
2.3 Order types	9
2.4 Market hours	9
2.5 Accounts, margin, and day-trading rules	9
2.6 Taxes, briefly	10
03 How a Trading Bot Works	11
3.1 The six components	11
3.2 Where AI fits	12
3.3 The life of one decision	12
04 Set Up Your Workshop	13
4.1 Install Python	13
4.2 Create the project	14
4.3 Open a paper-trading account	14
05 Your First API Calls	16
5.1 Connect and read the account	16
5.2 Download price history	17
5.3 Place and cancel a paper order	18
06 Designing a Simple Strategy	19
6.1 Three families	19
6.2 The starter strategy	19
6.3 Writing rules precisely	20
6.4 What not to do	20

07 Backtesting Without Fooling Yourself	21
7.1 A simple backtest	22
7.2 Metrics that matter	22
7.3 The seven traps	23
7.4 Out-of-sample testing	23
08 Building the Bot	24
8.1 Project layout	24
8.2 Configuration	24
8.3 The bot	25
8.4 Run it by hand first	27
8.5 Use an AI assistant to review it	27
09 Adding AI to the Decision	28
9.1 Three safe patterns	28
9.2 Example: a news veto	29
9.3 Guardrails for any AI component	29
10 Running It 100% Autonomously	30
10.1 Schedule it with cron	30
10.2 Run it in the cloud	31
10.3 Alerts	31
10.4 The kill switch	31
10.5 Daily supervision routine	31
11 Measuring Paper Results	32
11.1 How long	32
11.2 Trade journal	32
11.3 Go / no-go checklist	33
12 Going Live (Optional, and Small)	34
12.1 Scaling up	35
13 Troubleshooting	36
Quick Reference	37
Official resources	37
Glossary	38

Read This First: The Honest Truth

Before any code, you need a clear picture of what you're getting into. This chapter is short, but it's the most important one in the book.

IN THIS CHAPTER

- What “100% autonomous AI trading” actually means
- Why most bots lose money
- The rules this manual follows
- A self-check before you continue

⚠ REAL MONEY RISK

A fully autonomous bot places real orders with real money and no human checking each trade. A bug, bad data, an outage, or a strategy that stops working can drain an account in minutes. **Never connect a bot to money you can't afford to lose completely.**

1.1 What “autonomous” means here

An **autonomous trading system** is a program that, on its own and on a schedule:

1. **Gets market data**, such as prices and volumes.
2. **Decides** whether to buy, sell, or do nothing, using rules, a statistical model, or an AI model.
3. **Checks risk limits**, such as position size and daily loss.
4. **Sends orders** to a broker through an **API** (a doorway that lets programs talk to the broker).
5. **Logs everything** and alerts you when something unusual happens.

“100% autonomous” means steps 1–5 happen without you clicking anything. It does **not** mean unsupervised. Professionals running automated systems still monitor them, set kill switches, and review results every day.

1.2 Why most bots lose money

REASON	WHAT IT MEANS
Markets are competitive	You're up against firms with faster data, better models, and lower costs. Simple patterns that worked in the past tend to get “arbitraged away.”
Costs add up	Spreads, slippage, fees, and taxes eat into small edges. A strategy that looks profitable before costs is often a loser after them.
Overfitting	It's easy to tune a strategy until it looks perfect on old data. That usually means it memorised noise and will fail on new data.
Regime change	A strategy that worked in a calm, rising market can break in a crash or a choppy sideways market.
Bugs	A typo can mean buying 1,000 shares instead of 10, or never selling at all.
AI hallucination	Language models can confidently produce wrong numbers, invent news, or misread data.

⚠ BEWARE OF “GUARANTEED” AI BOTS

Regulators, including the SEC and FINRA, have repeatedly warned investors about scams that promise high or guaranteed returns from “AI” trading. No legitimate product guarantees profits. If someone sells you a bot with a screenshot of huge returns, assume it's marketing or fraud until you've proven otherwise. Check people and firms at brokercheck.finra.org and read the alerts at investor.gov.

1.3 The rules this manual follows

- ☐ Paper trade first, for at least several weeks of market days and many trades
- ☐ Every bot has a hard maximum position size
- ☐ Every bot has a daily loss limit that stops it automatically
- ☐ Every bot has a kill switch you can trigger in seconds
- ☐ Every decision and order is logged
- ☐ No leverage, options, or shorting until you fully understand them, and not in this book's starter bot
- ☐ API keys are kept secret and never pasted into chats, code, or screenshots
- ☐ Going live starts with a tiny amount of money you've already mentally written off

1.4 Self-check

Answer honestly. If you answer “no” to any of these, stay in paper trading.

- ☐ I can lose every dollar I put in without it affecting rent, bills, or debt payments
- ☐ I have an emergency fund and no high-interest debt
- ☐ I understand that simulated profits often don't survive real trading
- ☐ I'm doing this to learn, not to replace an income
- ☐ I'll check on the bot every trading day

CHAPTER 2

Markets in Plain English

You can't supervise a bot if you don't understand what it's doing. This chapter covers the minimum market knowledge you need.

IN THIS CHAPTER

- Stocks, ETFs, and crypto
- How a trade actually happens
- Order types
- Market hours
- Accounts, margin, and the new US intraday margin rules

2.1 What you can trade

ASSET	WHAT IT IS	BEGINNER NOTES
Stock	A small piece of ownership in a company	Individual stocks can swing a lot on news
ETF	A fund that holds many stocks and trades like one stock	Broad ETFs are a gentler place to learn
Crypto	Digital tokens such as Bitcoin	Trades 24/7, is very volatile, and has fewer investor protections
Options	Contracts that give the right to buy or sell later	Leveraged and complex. Not covered by the starter bot
Futures and forex	Leveraged contracts on commodities, indexes, and currencies	Not for beginners. Not covered here

2.2 How a trade happens

When your bot “buys 5 shares,” it sends an **order** to your **broker**. The broker routes it to a market where it’s matched with a seller. The **bid** is the highest price buyers are offering right now. The **ask** is the lowest price sellers will accept. The gap between them is the **spread**. When you buy at the ask and sell at the bid, you pay the spread every time.

Slippage is the difference between the price you expected and the price you actually got. It grows with fast markets, thinly traded assets, and large orders.

2.3 Order types

ORDER	WHAT IT DOES	WHEN TO USE IT
Market	Buy or sell now, at whatever price is available	Highly liquid assets, small sizes
Limit	Buy at this price or lower, or sell at this price or higher	You care about price more than speed
Stop	Becomes a market order once a trigger price is hit	Basic stop-loss
Stop-limit	Becomes a limit order once triggered	Price control, but it might not fill
Trailing stop	The stop price follows the market up (for a long position)	Protecting gains
Bracket	An entry with an attached take-profit and stop-loss	Bots that need automatic exits

Time in force says how long an order stays active. **DAY** expires at the close. **GTC** (good till cancelled) stays until it fills or you cancel it.

2.4 Market hours

The main US stock session runs 9:30 a.m. to 4:00 p.m. Eastern Time on trading days. Many brokers also offer pre-market and after-hours sessions, where spreads are usually wider. Markets are closed on weekends and exchange holidays. Your bot should check the market **clock** rather than guess. Crypto trades all day, every day.

2.5 Accounts, margin, and day-trading rules

A **cash account** lets you trade only with money you’ve deposited. A **margin account** lets you borrow from the broker, which magnifies both gains and losses.

☆ US DAY-TRADING RULES CHANGED IN 2026

For years, FINRA's "pattern day trader" rule required **\$25,000** in a margin account to make four or more day trades in five business days. On April 14, 2026, the SEC approved FINRA amendments to Rule 4210 that **replace** the pattern-day-trader rules and the \$25,000 minimum with new **intraday margin standards**. The amendments took effect June 4, 2026, but brokers may phase them in until **October 20, 2027**. That means your broker might still apply the old rules for now. Ask your broker which rules apply to your account. Source: [FINRA Regulatory Notice 26-10](#).

Cash accounts have their own limit: you have to wait for sale proceeds to **settle** before you reuse them, or you risk violations. US stock trades now settle in one business day (T+1).

2.6 Taxes, briefly

In the US, profits from assets held a year or less are usually taxed as **short-term capital gains**, at ordinary income rates. A bot can generate hundreds of taxable trades. The **wash-sale rule** can disallow a loss if you buy the same or a "substantially identical" security within 30 days before or after selling it at a loss, and bots trip this constantly. Keep your broker's tax forms and talk to a tax professional. Rules differ outside the US.

How a Trading Bot Works

Every automated system, from a hedge fund's to yours, has the same parts. Learn them once and every tutorial will make sense.

IN THIS CHAPTER

- The six components of a bot
- Where “AI” fits, and where it doesn't
- The life of a single decision

3.1 The six components

COMPONENT	JOB	IN OUR STARTER BOT
Data feed	Gets prices, volumes, and maybe news	Broker's market-data API
Strategy	Turns data into a signal: buy, sell, or hold	A simple moving-average rule
Risk manager	Can veto or shrink any trade	Position cap, daily loss limit, kill switch
Execution	Turns approved signals into orders	Broker trading API
Scheduler	Runs the loop at set times	<code>cron</code> or a cloud scheduler
Monitoring	Logs, alerts, daily reports	Log file plus an email or phone alert

3.2 Where AI fits

“AI trading” can mean very different things:

APPROACH	WHAT IT IS	HONEST TAKE
Rules written with AI help	You use an assistant like Claude or ChatGPT to <i>write</i> a rules-based bot	The most practical, transparent starting point. This book's main path
Machine-learning model	A model trained on historical data to predict moves	Very prone to overfitting. Needs careful validation
LLM as decision maker	A language model reads data or news and says buy or sell	Can be inconsistent and can hallucinate. If you use it, put hard rules around it
LLM as analyst	AI summarises news or flags unusual events for <i>you</i> or for rules to consider	Lower risk. A useful helper

TIP

The safest design is “**AI proposes, rules dispose.**” Let the AI suggest, but have plain code with hard limits decide what’s actually allowed.

3.3 The life of one decision

- 1 The scheduler wakes the bot**, for example at 3:45 p.m. ET on a trading day.
- 2 The bot checks the market clock.**
If the market is closed, it logs that and exits.
- 3 It checks the kill switch.**
If the switch is on, it exits immediately.
- 4 It pulls recent prices** for its watch list.
- 5 The strategy computes a signal** for each symbol.
- 6 The risk manager checks** current positions, today’s profit and loss, and the size limits.
- 7 Execution sends the approved orders**, usually limit orders.
- 8 It logs everything** and sends you a summary.

Set Up Your Workshop

Now you'll install the tools. You'll need about an hour.

IN THIS CHAPTER

- Install Python
- Create a project folder and virtual environment
- Open a free paper-trading account
- Store API keys safely

4.1 Install Python

1 **Download Python 3** from python.org/downloads. On Windows, tick **Add python.exe to PATH** during setup.

2 **Check it works.**

Open a terminal (see Vol. 1, Chapter 2) and type:

TERMINAL

```
python3 --version
```

On Windows the command may be `python` or `py`.

3 **Install an editor.**

VS Code or Cursor are both fine.

4.2 Create the project

TERMINAL

```
mkdir trading-bot && cd trading-bot
python3 -m venv .env
source .env/bin/activate      # Windows: .env\Scripts\activate
pip install alpaca-py pandas python-dotenv
```

A **virtual environment** (`.env`) keeps this project's packages separate from everything else on your computer. Activate it every time you open a new terminal for the project.

4.3 Open a paper-trading account

This book uses **Alpaca** as its example because it offers a free paper-trading environment with the same API as live trading. According to Alpaca's docs, anyone can sign up for a paper-only account with just an email address, and a new paper account starts with a simulated \$100,000 balance. Other brokers, such as Interactive Brokers, also offer paper accounts and APIs. This isn't a recommendation. Compare brokers yourself.

- 1 **Sign up** at alpaca.markets and choose the paper-only option.
- 2 **Open the paper dashboard** and find **API Keys**.
- 3 **Generate a key pair.**
You'll see a **Key ID** and a **Secret Key**. The secret is usually shown only once.
- 4 **Save them in a `.env` file** in your project folder (below).

.ENV

```
APCA_API_KEY_ID=your-paper-key-id
APCA_API_SECRET_KEY=your-paper-secret
TRADING_MODE=paper
```

.GITIGNORE

```
.env
.venv/
logs/
```

 PROTECT YOUR KEYS

Anyone with your API keys can trade your account. Never commit `.env` to GitHub, paste keys into an AI chat, or show them on screen. If a key leaks, delete it in the dashboard right away and make a new one.

 PAPER ISN'T REALITY

Alpaca's own documentation lists what paper trading doesn't simulate: market impact, slippage due to latency, queue position, price improvement, regulatory fees, and dividends. It also fills orders regardless of the real size available. Expect live results to be **worse** than paper.

Your First API Calls

Before building anything automatic, prove you can talk to the broker by hand.

IN THIS CHAPTER

- Connect to the paper account
- Read your balance and the market clock
- Download price history
- Place, view, and cancel a paper order

5.1 Connect and read the account

CHECK_ACCOUNT.PY

```
import os
from dotenv import load_dotenv
from alpaca.trading.client import TradingClient

load_dotenv()
client = TradingClient(
    os.environ["APCA_API_KEY_ID"],
    os.environ["APCA_API_SECRET_KEY"],
    paper=True,  # ALWAYS True while learning
)

acct = client.get_account()
print("Status:", acct.status)
print("Cash:", acct.cash)
print("Equity:", acct.equity)

clock = client.get_clock()
print("Market open now?", clock.is_open, "| next open:", clock.next_open)
```

Run it with `python check_account.py`. If you see your balance, you're connected.

5.2 Download price history

GET_BARS.PY

```
import os
from datetime import datetime, timedelta
from dotenv import load_dotenv
from alpaca.data.historical import StockHistoricalDataClient
from alpaca.data.requests import StockBarsRequest
from alpaca.data.timeframe import TimeFrame

load_dotenv()
data = StockHistoricalDataClient(os.environ["APCA_API_KEY_ID"],
                                os.environ["APCA_API_SECRET_KEY"])
req = StockBarsRequest(symbol_or_symbols="SPY",
                       timeframe=TimeFrame.Day,
                       start=datetime.now() - timedelta(days=400))
bars = data.get_stock_bars(req).df
print(bars.tail())
bars.to_csv("spy_daily.csv")
```

Each row is a **bar**: open, high, low, close, and volume for one day.

5.3 Place and cancel a paper order

PAPER_ORDER.PY

```
import os
from dotenv import load_dotenv
from alpaca.trading.client import TradingClient
from alpaca.trading.requests import LimitOrderRequest
from alpaca.trading.enums import OrderSide, TimeInForce

load_dotenv()
client = TradingClient(os.environ["APCA_API_KEY_ID"],
                      os.environ["APCA_API_SECRET_KEY"], paper=True)

order = client.submit_order(LimitOrderRequest(
    symbol="SPY", qty=1, side=OrderSide.BUY,
    time_in_force=TimeInForce.DAY,
    limit_price=1.00,           # far below market, so it won't fill
))
print("Submitted:", order.id, order.status)

for o in client.get_orders():
    print(o.symbol, o.side, o.qty, o.status)

client.cancel_order_by_id(order.id)
print("Cancelled.")
```

TIP

Using a limit price far from the market is a safe way to practise the order lifecycle without getting filled.

Designing a Simple Strategy

A **strategy** is a precise rule you could write on an index card. If you can't explain it in two sentences, it's too complicated for your first bot.

IN THIS CHAPTER

- Three beginner strategy families
- The starter strategy: moving-average trend filter
- Writing rules precisely
- What not to do

6.1 Three families

FAMILY	IDEA	EXAMPLE RULE
Trend following	Things moving up tend to keep moving up for a while	Hold SPY only when price is above its 200-day average
Mean reversion	Big short-term moves tend to partly reverse	Buy after an unusually large 3-day drop, sell after a bounce
Rebalancing	Hold fixed weights and trade back to them	60% stock ETF / 40% bond ETF, rebalanced monthly

6.2 The starter strategy

We'll build a **moving-average trend filter** on one broad ETF. It trades rarely, which keeps costs and taxes low, and it's easy to check by hand.

Rule: Once per trading day, near the close:

- If yesterday's close is **above** the 200-day simple moving average and we hold nothing → **buy** a fixed dollar amount.
- If yesterday's close is **below** the 200-day average and we hold shares → **sell** everything.
- Otherwise → do nothing.

This isn't a recommendation and isn't claimed to be profitable. It's a teaching example because it's transparent and easy to test.

6.3 Writing rules precisely

Vague: "buy when it's going up." Precise: "At 3:45 p.m. ET, if the last completed daily close is greater than the mean of the previous 200 daily closes, and position = 0, buy floor($\$1,000 \div$ last price) shares with a DAY limit order at last price + 0.1%."

Ask an AI assistant to help:

PROMPT

Here is my trading idea in plain English: [idea]. Rewrite it as precise, unambiguous rules: exact data used, exact time, entry condition, exit condition, position size, order type, and what happens when data is missing. List any ambiguities you had to resolve.

6.4 What not to do

- ☐ Don't add indicators until the backtest looks great. That's overfitting
- ☐ Don't trade dozens of symbols at first
- ☐ Don't trade options, leverage, or penny stocks with a bot
- ☐ Don't let an AI change the strategy live without review
- ☐ Don't "average down" automatically into a losing position

Backtesting Without Fooling Yourself

A **backtest** runs your rules on historical data to see what *would* have happened. It's useful, and it's also the easiest place to lie to yourself.

IN THIS CHAPTER

- A complete, simple backtest in Python
- Metrics that matter
- The seven backtesting traps
- Out-of-sample testing

7.1 A simple backtest

BACKTEST.PY

```
import pandas as pd

df = pd.read_csv("spy_daily.csv")
df = df[["timestamp", "close"]].copy()
df["sma200"] = df["close"].rolling(200).mean()
# Decide on yesterday's data, act today, so there's no look-ahead
df["signal"] = (df["close"].shift(1) > df["sma200"].shift(1)).astype(int)
df["ret"] = df["close"].pct_change()
cost = 0.0005 # 0.05% per trade, a guess
trades = df["signal"].diff().abs().fillna(0)
df["strat"] = df["signal"] * df["ret"] - trades * cost
df = df.dropna()

eq = (1 + df["strat"]).cumprod()
bh = (1 + df["ret"]).cumprod()
dd = (eq / eq.cummax() - 1).min()
print(f"Strategy growth: {eq.iloc[-1]:.2f}x | Buy & hold: {bh.iloc[-1]:.2f}x")
print(f"Max drawdown: {dd:.1%} | Trades: {int(trades.sum())}")
```

You'll need years of data for a 200-day average to mean anything. Change `timedelta(days=400)` in `get_bars.py` to cover many more years if your data plan allows it.

7.2 Metrics that matter

METRIC	MEANING	WHY IT MATTERS
Total return	How much the account grew	Compare it with simply buying and holding
Max drawdown	Worst peak-to-trough fall	Could you stomach it? Could the bot survive it?
Number of trades	How often it trades	More trades means more costs and taxes
Win rate	Percentage of profitable trades	Misleading on its own. Many winning systems have low win rates
Sharpe ratio	Return per unit of volatility	Higher is better. Be suspicious of very high values
Exposure	Percentage of time invested	Low exposure can make returns look safer than they are

7.3 The seven traps

TRAP	WHAT GOES WRONG	DEFENCE
Look-ahead bias	Using information you wouldn't have had yet	<code>shift(1)</code> , so you decide on yesterday and act today
Overfitting	Tuning until it fits the past perfectly	Few parameters. Test on unseen data
Survivorship bias	Testing only today's winners	Use broad index ETFs, or data that includes delisted stocks
Ignoring costs	Assuming free, perfect fills	Add spread, slippage, and fees. Then double them
Data errors	Splits, gaps, bad ticks	Use adjusted data and plot it to check
Too little data	A few months of one market mood	Cover multiple years, including crashes
Many tries	Testing 100 ideas and keeping the best	Count every idea you tried. The "best" is often luck

7.4 Out-of-sample testing

Split your history: develop on the **first** 70%, then run the finished rules **once** on the last 30%. If it falls apart there, the idea probably doesn't work. Don't go back and tweak it until the test set looks good. That turns the test set into training data.

CHAPTER 8

Building the Bot

Now combine everything into one program that can run unattended **in paper mode**.

IN THIS CHAPTER

- Project layout
- The full starter bot, with risk checks
- Logging
- A dry-run mode

8.1 Project layout

TEXT

```
trading-bot/  
├ .env           # keys (never committed)  
├ config.py      # all limits in one place  
├ bot.py         # the main loop  
├ KILL_SWITCH    # create this file to stop the bot  
└ logs/
```

8.2 Configuration

CONFIG.PY

```
SYMBOL = "SPY"  
DOLLARS_PER_TRADE = 1000    # max dollars per entry  
MAX_POSITION_DOLLARS = 1500 # hard cap on total exposure  
MAX_DAILY_LOSS = 100       # stop trading if today's loss exceeds this  
SMA_DAYS = 200  
DRY_RUN = True             # True = log decisions, send NO orders  
PAPER = True               # must be True until Chapter 12
```

8.3 The bot

BOT.PY

```
import os, sys, logging, math
from datetime import datetime, timedelta
from pathlib import Path
from dotenv import load_dotenv
from alpaca.trading.client import TradingClient
from alpaca.trading.requests import LimitOrderRequest
from alpaca.trading.enums import OrderSide, TimeInForce
from alpaca.data.historical import StockHistoricalDataClient
from alpaca.data.requests import StockBarsRequest
from alpaca.data.timeframe import TimeFrame
import config as C

Path("logs").mkdir(exist_ok=True)
logging.basicConfig(filename=f"logs/{datetime.now():%Y-%m-%d}.log",
                    level=logging.INFO,
                    format="%(asctime)s %(levelname)s %(message)s")
log = logging.info

def main():
    load_dotenv()
    if Path("KILL_SWITCH").exists():
        log("Kill switch present. Exiting."); return
    if not C.PAPER or os.environ.get("TRADING_MODE") != "paper":
        log("Refusing to run: not in paper mode."); return

    key, sec = os.environ["APCA_API_KEY_ID"], os.environ["APCA_API_SECRET_KEY"]
    tc = TradingClient(key, sec, paper=True)
    dc = StockHistoricalDataClient(key, sec)
```

BOT.PY (CONTINUED)

```

if not tc.get_clock().is_open:
    log("Market closed. Exiting."); return

acct = tc.get_account()
day_pl = float(acct.equity) - float(acct.last_equity)
if day_pl < -C.MAX_DAILY_LOSS:
    log(f"Daily loss limit hit ({day_pl:.2f}). Exiting."); return

bars = dc.get_stock_bars(StockBarsRequest(
    symbol_or_symbols=C.SYMBOL, timeframe=TimeFrame.Day,
    start=datetime.now() - timedelta(days=C.SMA_DAYS * 2))).df
closes = bars["close"]
if len(closes) < C.SMA_DAYS + 1:
    log("Not enough data. Exiting."); return
last, sma = closes.iloc[-1], closes.iloc[-C.SMA_DAYS:].mean()

pos = next((p for p in tc.get_all_positions() if p.symbol == C.SYMBOL), None)
held = float(pos.market_value) if pos else 0.0
log(f"last={last:.2f} sma={sma:.2f} held=${held:.2f} day_pl={day_pl:.2f}")

if last > sma and held == 0:
    qty = math.floor(min(C.DOLLARS_PER_TRADE, C.MAX_POSITION_DOLLARS) / last)
    if qty < 1:
        log("Qty < 1, skip."); return
    place(tc, OrderSide.BUY, qty, round(last * 1.001, 2))
elif last < sma and pos:
    place(tc, OrderSide.SELL, int(float(pos.qty)), round(last * 0.999, 2))
else:
    log("No action.")

```

BOT.PY (CONTINUED)

```

def place(tc, side, qty, limit):
    msg = f"{side.value} {qty} {C.SYMBOL} @ {limit}"
    if C.DRY_RUN:
        log("DRY RUN, would place: " + msg); return
    o = tc.submit_order(LimitOrderRequest(symbol=C.SYMBOL, qty=qty, side=side,
        time_in_force=TimeInForce.DAY, limit_price=limit))
    log(f"Placed {msg} id={o.id}")

if __name__ == "__main__":
    try:
        main()
    except Exception as e:
        logging.exception("Bot crashed: %s", e)
        sys.exit(1)

```

i NOTE

The daily bars include today's bar while the market is open, so `last` is roughly the current price. That's acceptable for a near-close daily rule, but now you know exactly what the code does. Understanding details like this is your job as the supervisor.

8.4 Run it by hand first

- 1 Keep `DRY_RUN = True`.**
Run `python bot.py` during market hours.
- 2 Read the log in `logs/`.** Check `last` and `sma` against a chart.
- 3 Test each safety path.**
Create the `KILL_SWITCH` file and confirm the bot exits. Set `MAX_DAILY_LOSS = -1` and confirm it refuses to trade. Run it while the market is closed.
- 4 Set `DRY_RUN = False`** (still paper) and confirm a paper order appears in the dashboard.

8.5 Use an AI assistant to review it

PROMPT

Review this trading bot for bugs that could lose money: wrong quantities, missing checks, wrong side, repeated orders, look-ahead bias, unhandled errors, or anything that could run in live mode by accident. Don't add features. List issues by severity, then propose minimal fixes.

Adding AI to the Decision

This chapter shows how to add an AI model **safely**, as a filter or analyst inside hard limits, never as an unrestricted trader.

IN THIS CHAPTER

- Three safe patterns
- Example: an LLM “news veto”
- Guardrails for any AI component

9.1 Three safe patterns

PATTERN	HOW IT WORKS	RISK LEVEL
AI writes, human approves	AI drafts strategy code; you review and backtest it	Lowest
AI veto	Rules propose a trade; AI may only say “skip” for unusual news	Low: it can only <i>reduce</i> trading
AI sizing within caps	AI suggests 0–100% of a capped size	Medium
AI full control	AI decides what, when, and how much	Highest. Not recommended

9.2 Example: a news veto

The idea: before buying, ask a language model whether recent headlines describe an extraordinary event. If its answer isn't a clean "OK," skip the trade. A bad answer can only make you trade **less**.

VETO.PY

```
def ai_allows_trade(headlines: list[str], ask_llm) -> bool:
    prompt = ("You are a cautious risk checker. Reply with exactly one word: "
              "OK or SKIP. Reply SKIP only if these headlines describe an "
              "extraordinary market event (e.g. trading halt, crash, war "
              "escalation, fraud at the company).\n\n" + "\n".join(headlines))

    try:
        reply = ask_llm(prompt).strip().upper()
    except Exception:
        return False          # on any error, don't trade
    return reply == "OK"
```

`ask_llm` is whatever function calls your chosen AI provider's API. Every provider's API is different, and most charge per use, so check current pricing and documentation for the one you choose.

9.3 Guardrails for any AI component

- ☐ AI output is parsed strictly. Anything unexpected means "no trade"
- ☐ AI can never raise the hard limits in `config.py`
- ☐ AI never sees or handles API keys
- ☐ Every AI prompt and reply is logged
- ☐ You backtest or paper-test the AI version against the plain version
- ☐ If the AI provider is down, the bot fails safe and doesn't trade
- ☐ Watch for prompt injection: news text can contain instructions meant to manipulate an AI

Running It 100% Autonomously

Now the bot runs on a schedule with no clicks from you, still in **paper** mode.

IN THIS CHAPTER

- Scheduling with cron (Mac and Linux) or Task Scheduler (Windows)
- Running in the cloud
- Alerts and the kill switch
- A daily supervision routine

10.1 Schedule it with cron

`cron` runs commands on a timetable. Type `crontab -e` and add:

TERMINAL

```
# 3:45 pm, Monday-Friday. Cron uses the computer's local time zone!  
45 15 * * 1-5 cd /home/you/trading-bot && .venv/bin/python bot.py
```

⚠ TIME ZONES

If your computer isn't on US Eastern Time, adjust the hour. Daylight-saving changes can shift things too. The bot checks `is_open` itself, but a wrong hour can mean it runs after the close and silently does nothing.

On Windows, use **Task Scheduler**: create a basic task, run it weekly on Mon–Fri at the chosen time, and set the action to start `.venv\Scripts\python.exe` with the argument `bot.py`, starting in your project folder.

10.2 Run it in the cloud

A laptop that sleeps will miss runs. Options include a small cloud virtual machine (a **VPS**), or a scheduled job on a cloud platform. AWS, Google Cloud, Azure, and smaller hosts all offer these. On a cloud machine:

- ☐ Store keys in the platform's secret manager or a protected `.env`, never in code
- ☐ Turn on automatic security updates
- ☐ Lock down remote login with SSH keys, not passwords
- ☐ Set up a billing alert so a mistake can't run up a cloud bill

10.3 Alerts

Have the bot email or message you after every run, and especially on errors. Simple options include email through your provider's SMTP, or a push service. At minimum, write a daily summary line to the log and check it.

10.4 The kill switch

Create a file named `KILL_SWITCH` in the project folder and the bot stops at its next run. For an emergency, also:

EMERGENCY STOP

- 1 Create `KILL_SWITCH`.
- 2 Cancel all open orders in the broker dashboard.
- 3 Close positions manually if you need to.
- 4 Disable or delete the API key in the dashboard. This stops *everything* using that key.
- 5 Disable the cron job (`crontab -e`, then put `#` in front of the line).

10.5 Daily supervision routine

- ☐ Read today's log. Any errors or warnings?
- ☐ Do the positions in the dashboard match what the bot thinks it holds?
- ☐ Is the day's P&L within expectations?
- ☐ Did the scheduled run actually happen?
- ☐ Weekly: compare paper results with the backtest's expectations

Measuring Paper Results

Paper trading is your rehearsal. Here's how to judge it honestly.

IN THIS CHAPTER

- How long to paper trade
- What to record
- Go/no-go criteria

11.1 How long

There's no magic number. A slow strategy like the starter bot might only trade a few times a year, so a few weeks of paper trading mostly tests whether the **plumbing** works, not whether the strategy works. For faster strategies, aim for many trades across different market conditions. In every case, run long enough to see at least one stressful market week.

11.2 Trade journal

DATE	SIGNAL	ORDER	FILL PRICE	EXPECTED PRICE	SLIPPAGE	NOTES

11.3 Go / no-go checklist

- ☐ Zero unexplained errors for the last several weeks of runs
- ☐ Every safety path (kill switch, loss limit, market closed) tested and working
- ☐ Paper behaviour matches the backtest logic trade-for-trade
- ☐ I understand every line of the code, or had it reviewed and asked about every part I didn't
- ☐ I accept that live results will probably be worse than paper
- ☐ I have decided the maximum I'm willing to lose and can afford to lose it

Going Live (Optional, and Small)

Going live is a choice, not the goal of this book. Many readers will, and should, stay in paper mode indefinitely.

IN THIS CHAPTER

- The small-money rule
- Switching safely
- Scaling up, or not

⚠ THIS IS WHERE REAL LOSSES HAPPEN

Live trading involves real money, real taxes, and real emotional pressure. A bot that worked perfectly on paper can behave differently live because of fills, slippage, and partial executions. Start with an amount you've already accepted losing.

SWITCH TO LIVE SAFELY

- 1 Open and fund a live brokerage account** with a small amount.
- 2 Create separate live API keys.**
Never reuse paper keys, and never mix them up.
- 3 Lower every limit.**
For example, set `DOLLARS_PER_TRADE` to a fraction of your paper size.
- 4 Change the code deliberately.**
Set `paper=False`, `PAPER = False`, and `TRADING_MODE=live` in three places, so it can't happen by accident.

- 5 **Run once by hand with** `DRY_RUN = True` **against the live account.**
- 6 **Enable real orders and watch the first live trade** in the dashboard.
- 7 **Keep the daily routine.**
If anything surprises you, use the kill switch first and investigate after.

12.1 Scaling up

Only increase size after a long live record that matches your expectations. Increase gradually. Re-check that limits, alerts, and the kill switch still match the new size. And keep the rest of your savings in investments you'd be comfortable holding without a bot.

Troubleshooting

PROBLEM	LIKELY CAUSE	FIX
<code>KeyError: 'APCA_API_KEY_ID'</code>	<code>.env</code> not loaded, or the wrong folder	Run from the project folder. Check the variable names
401 / unauthorized	Wrong keys, or live keys used with <code>paper=True</code>	Regenerate paper keys. Check the mode
403 / forbidden on an order	Insufficient buying power, or the asset isn't tradable	Check account cash and the asset's status
Order never fills	Limit price too far from the market, or market closed	Check the quote and the market clock
Bot buys twice	No check for existing open orders	Check <code>get_orders()</code> before submitting
Nothing happens at the scheduled time	Cron time zone, venv path, or the computer was asleep	Log the start of every run. Use absolute paths. Use a cloud VM
Empty data	Symbol typo, market holiday, or data-plan limits	Print the request. Check the symbol
Backtest great, paper bad	Overfitting or unrealistic costs	Simplify. Add costs. Re-test out-of-sample
AI component gives odd answers	Ambiguous prompt, or model variability	Strict output format. Default to "no trade"

Quick Reference

Safety first: paper mode → dry run → paper orders → weeks of supervision → (optional) tiny live.

Hard limits in config.py: `DOLLARS_PER_TRADE` , `MAX_POSITION_DOLLARS` , `MAX_DAILY_LOSS` , `DRY_RUN` , `PAPER` .

Emergency: create `KILL_SWITCH` → cancel orders → close positions → revoke API key → disable cron.

Daily: read log · positions match · P&L within limits · run happened.

Never: commit keys · give AI the keys · use leverage in the starter bot · trust a “guaranteed” bot.

Official resources

RESOURCE	ADDRESS
Investor education and alerts (SEC)	investor.gov
Check brokers and advisers (FINRA)	brokercheck.finra.org
FINRA intraday margin notice 26-10	finra.org/rules-guidance/notices/26-10
Alpaca paper trading docs	docs.alpaca.markets/docs/paper-trading
Python downloads	python.org/downloads

Glossary

API	A doorway that lets programs talk to a service, such as a broker, without a human clicking.
API key	A secret pair of codes that lets your program access your account. Treat it like a password.
Ask	The lowest price a seller currently accepts.
Backtest	Running a strategy's rules on historical data to see what would have happened.
Bar	One time period's open, high, low, close, and volume.
Bid	The highest price a buyer is currently offering.
Bracket order	An entry order with an automatic take-profit and stop-loss attached.
Cash account	A brokerage account that trades only with deposited money, with no borrowing.
Cron	A scheduler on Mac and Linux that runs commands at set times.
Drawdown	The fall from an account's peak value to a later low.
Dry run	Running the bot so it logs what it <i>would</i> do without sending orders.
ETF	Exchange-traded fund: a basket of securities that trades like one stock.
Kill switch	A fast, reliable way to stop a bot from trading.
Limit order	An order to trade only at a set price or better.
Look-ahead bias	A backtest error where the strategy uses information it wouldn't have had at that time.
Margin	Money borrowed from a broker to trade. It magnifies gains and losses.
Market order	An order to trade immediately at the best available price.

Moving average	The average closing price over the last N periods, used to smooth price data.
Out-of-sample	Data held back and not used while developing a strategy, used once for an honest test.
Overfitting	Tuning a strategy so closely to past data that it fails on new data.
Paper trading	Simulated trading with fake money using real market prices.
Pattern day trader	A former US FINRA designation for margin accounts making four or more day trades in five business days, being replaced by intraday margin standards in 2026–2027.
Position	How much of an asset you currently hold.
Prompt injection	Text hidden in data, such as a news article, that tries to manipulate an AI's instructions.
Settlement	When a trade's cash and securities officially change hands. US stocks settle T+1.
Sharpe ratio	A measure of return relative to volatility.
Slippage	The difference between the expected price and the actual fill price.
Spread	The gap between the bid and the ask.
Stop order	An order that becomes active once a trigger price is reached.
Virtual environment	An isolated folder of Python packages for one project.
Wash sale	A US tax rule that can disallow a loss if you buy a substantially identical security within 30 days before or after the sale.