Trading Boosters · Nov 19, 2025 · 8 min read

Intraday Volatility Breakout Blueprint

This guide shows how to build and test simple breakout systems using free Python tools, Alpaca data, and modular research steps.

Peter

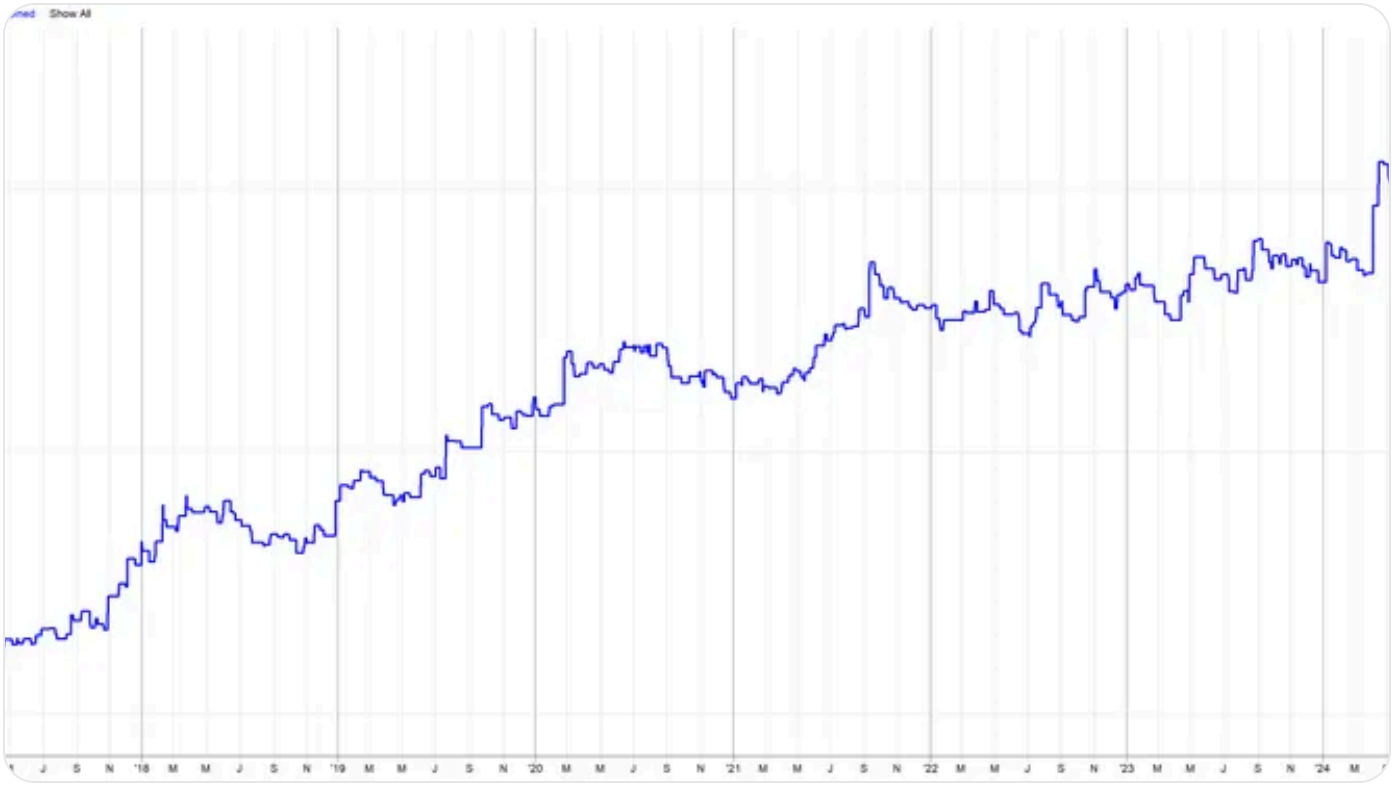


TABLE OF CONTENTS +

Intraday breakout strategies are simple in design but powerful in practice. The logic is easy to understand, works across many markets, and can be adapted to different volatility regimes.

How the logic of a volatility breakout works is explained in the earlier post Day Trading Volatility Breakouts Systematically [All Rules Included]

But as explained in the original post, simple breakout logic shows strong potential but also produces many trades. When these trades happen in a low volatility environment with a low chance of a trend day, fees and slippage can reduce performance. Fortunately, we do not have to use only the simple breakout logic itself. We can choose to trade breakouts only in favorable environments where the odds of a trend day are higher.

To find such conditions, we need to be able to backtest and validate our ideas. In this article I show you how I do it using free tools, simple Python scripts, and support from LLMs. The goal is not to create a complex trading framework, but to build a clear, modular workflow that you can extend step by step.

Why use Python and free data

Most traders test intraday strategies in commercial platforms like TradeStation, NinjaTrader or SierraChart. These tools are convenient and integrate already with data sources, but they usually require a funded brokerage account, paid licenses, or paid data subscriptions.

To keep this blueprint fully free, we will use Python. Python is not the easiest tool for non programmers. I am not a programmer myself. But today it has one huge advantage: it works very well with LLM coding support. Once you have a basic structure, like the one I share below, you can let ChatGPT or Claude modify your scripts and test new ideas with minimal work.

For data I use Alpaca.markets. Alpaca is a broker, and you must register to get access, but you do not need to fund your account. Even without funding you can download long history of high quality SIP intraday data for stocks and ETFs. Alpaca does not offer futures data, so in this blueprint we will use ETF versions of futures markets. The breakout logic behaves very similarly, and once you build a strategy you like you can retest it on your broker data for futures or microfutures.

You can download Alpaca data using their API or their official Python SDK: https://github.com/alpacahq/alpaca-py. Or you can use my own downloader, which saves 1 minute data into csv files. I share it below.

Core idea of intraday volatility breakout

Intraday breakout logic can be very simple. A basic version looks like this:

- calculate ATR as the average daily range
- set breakout levels at close -/+ k x ATR (for example 0.33 x ATR)
- enter long above the upper level or enter short below the lower level
- use a fixed small stop (for example 0.33 x ATR)
- exit at end of day

This approach trades frequently and is too simple for live use, but it is a perfect starting point. Once you test this basic version, you can add filters that improve context and reduce noise.

Simple breakout with the above rules applied to two indexes — SPY (S&P 500) and QQQ (Nasdaq 100). Commissions of $0.005 per share, with a $1 minimum per trade, are included. The performance is not bad, but the average trade size would be too small for live trading. You can replicate the same results with the script shared below.

A common improvement is to trade breakouts only after specific daily conditions, such as:

- a narrow day
- low volatility contraction
- a specific trend filter
- a volatility expansion pattern

Your logic becomes:

- if market context is OK -> set next day breakout levels

Equity of an intraday volatility breakout with a market-context filter applied. Still tested on two indexes — SPY (S&P 500) and QQQ (Nasdaq 100). Commissions of $0.005 per share, with a $1 minimum per trade, are included. The expectancy of a single trade is 3× higher, while the number of trades dropped 6×. This is very close to the version I trade live myself. You can find the market-context logic in the RealTest template I share below (the logic is explained and can be used even if you code the system in Python, etc.).

You can also adjust timing. Levels do not have to be based on the Close. You can use the next Open, or the first 30 minutes. You can include premarket behaviour. You can test different exits or allow multiple entries per day.

There are many possibilities, and most of them introduce the risk of over-optimization. This is why I always begin with a simple prototype and add complexity only when the base model is robust.

Two constants that help avoid mistakes and keep the backtester simple:

- limiting to one breakout attempt per market per day

- use a simple exit (fixed stop and end-of-day exit)

This reduces the chance of coding errors and makes the behaviour easier to validate.

Building a simple backtest workflow

You can build a complex backtesting framework that handles everything at once, but if you rely on LLM coding support, this is risky. LLMs can produce working code, but they also make small mistakes, and those mistakes can ruin your backtest.

My solution is to break the backtest into simple blocks. Each block does one job. When I change something in one block, nothing else breaks.

A clean workflow looks like this:

Data downloader

Downloads and stores 1 minute data.
Output: csv files.

Basic breakout engine

Runs one breakout attempt per day per symbol with fixed rules.
Output: csv with all trades, plus metadata like ATR.

Trade filter

Applies conditions such as narrow day or trend filter.
Output: filtered trades.

Portfolio builder

Combines trades from multiple markets.
Output: unified trade list.

Portfolio analysis

Creates charts and statistics.

Combining blocks into workflow

Each block is simple and easy to ask an LLM to modify. For example, if I want to change the breakout distance, I edit only block 2. All other blocks stay untouched.

For some parts of my workflow I might even use third-party software I already use in trading. For example RealTest that has build in functionality to import trades. It can import trades from Python and run filtering and portfolio analysis much faster than a custom script. But you can do everything in Python if you want.

Practical example and ready resources

If you want to build intraday breakout strategies the same way I do, I prepared a full set of ready-made resources. These files allow you to avoid the technical setup and move directly to real research. The goal is to give you a complete workflow that works even if you are not a programmer.

The scripts are available in the supporter section. After subscribing you will also get access to the comments section under the article. Please use that space for questions, not email. The fee is small, but it helps me limit the number of support requests I receive.

Below is a detailed description of what the resources include and how they fit into the intraday breakout workflow.

1) Data downloader (block 1)

You get a Python script that downloads high quality SIP 1 minute data from Alpaca (see Comparing Affordable Intraday Data Sources: TradeStation vs. Polygon vs. Alpaca). You only need a free Alpaca account. No funding and no fees are required.

The downloader lets you:

- select symbols (SPY, QQQ, IWM, GLD, TLT, sector ETFs, futures proxies, etc.)
- set the date range
- automatically save clean csv files
- maintain a simple directory structure compatible with the next steps

This saves hours of work and avoids the need to learn the Alpaca API from scratch.

Get the downloader here: [Downloading Stock/ETF Market Data with Python and Alpaca API]

2) Intraday breakout trade generator (block 2)

This is the core script. It reads the 1 minute data and executes a simple breakout logic:

- one breakout attempt per day
- volatility based breakout levels (ATR x multiplier)
- fixed stop loss
- end of day exit
- ATR and other metadata included
- output stored as csv files (one per symbol)

This script is intentionally simple and transparent. It produces a clean set of raw trades that are easy to review and easy to modify. You can adjust the ATR length, the breakout distance, the stop size, and other parameters. The README explains everything step by step. If you need to change something, you can ask ChatGPT or Claude to modify the code for you.

Blocks 1 and 2 give you all trades without any market filter. These raw trades are the base of the whole workflow.

Get the backtester here: [Intraday Breakout Strategy Backtester]

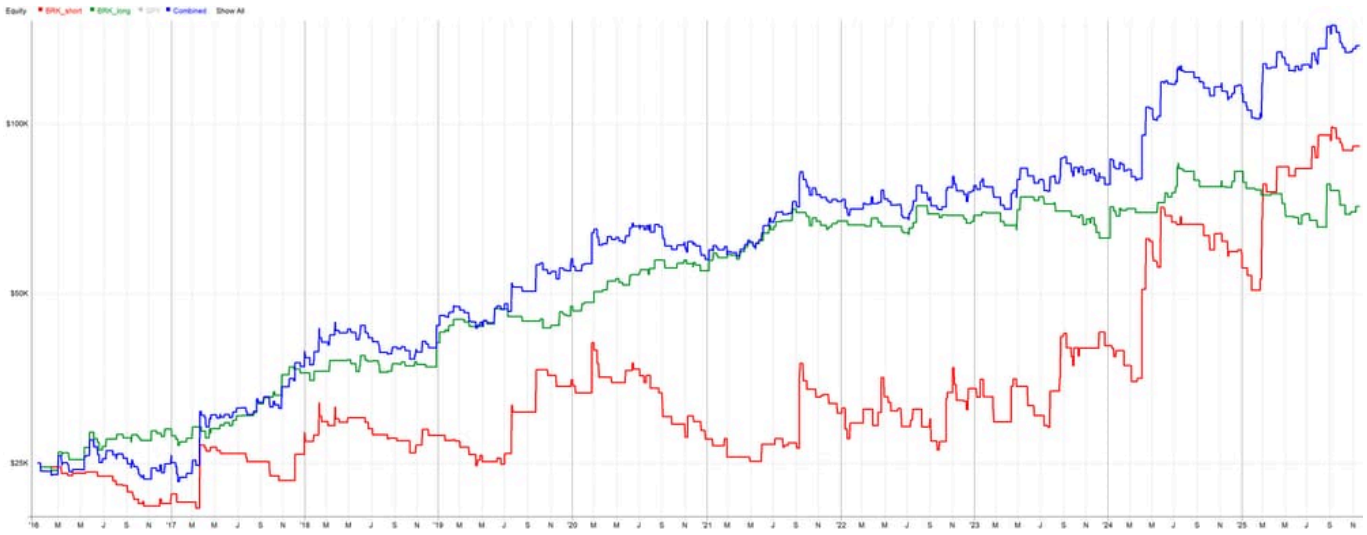
3 + 4 + 5 handled fully inside RealTest (filters, portfolio building, analytics)

Instead of writing extra Python scripts for filtering, portfolio construction, and performance analysis, I provide one consolidated RealTest script that handles **all three steps**.

This RealTest script:

- imports the raw trades generated by block 2
- applies market condition filters (narrow day, trend filter, volatility contraction, etc.)
- builds a combined portfolio from all markets
- controls position sizing

- generates full equity curves and detailed statistics
- allows you to switch filters and parameters quickly



Intraday breakouts are great portfolio diversifiers, as they can trade long or short based on the intraday trend. This is the equity curve with the context filter I share in the RealTest template. It's very close to what I trade myself and can be a good starting point for researching and learning intraday breakouts.

This setup gives you research-level functionality with very little effort. RealTest is extremely fast, reliable and transparent. It also eliminates the risk of Python coding mistakes that often happen when working with LLM generated code.

Your full workflow becomes:

1. download data (block 1)
2. generate breakout trades (block 2)
3. use RealTest to filter, combine and analyze (blocks 3 + 4 + 5)

This is the same approach I use for my own strategy development. It keeps everything simple, modular and safe.

Get the RealTest script here: Intraday Breakout Strategy - RealTest Analysis Script

Summary

Breakout strategies can be very simple and still work across many markets with the same settings. Do not focus only on a perfect looking equity curve. I prefer simple logic

with minimal variables that fit well into my whole portfolio. This is why I use RealTest for final portfolio backtesting and filtering. I test the intraday breakout together with my other strategies, like mean reversion and momentum, because I care mainly about overall portfolio performance.

Tools like NinjaTrader or TradeStation make it easy to backtest breakout logic on a single market. But many developers stop there and do not view results from a portfolio perspective. Instead, they add more and more variables to make the single market equity look great, which usually does not work in live trading.

I trade intraday breakouts in a very simple way. I use one daily market filter, and when the conditions are true, I submit the breakout levels for the next day. That is all.

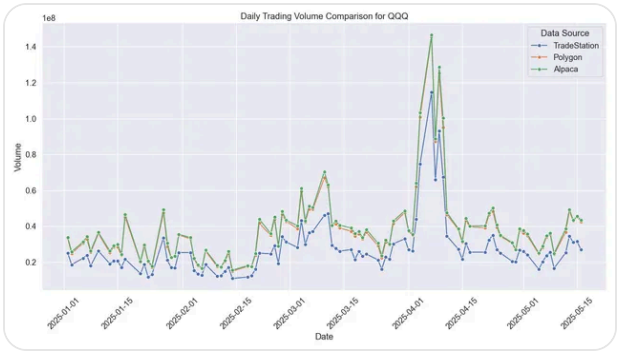
As always, improvement comes with practice. For that reason it makes sense to start very small, even with a weak setup, and use real market feedback (with very small risk) to guide your next backtests. This process will lead you toward a more efficient and robust solution.

Read next

Trading Boosters · Jan 5, 2026

Do Breakouts of Strong Swings Work? I Tested 40 Futures Markets and the Data Is Clear

Many traders spend hours hunting for strong support and resistance levels with the goal of entering counter-trend trades. They bet on price hitting a wall and reversing.



Trading Boosters · May 21, 2025

Comparing Affordable Intraday Data Sources: TradeStation vs. Polygon vs. Alpaca

Building intraday strategies? We compare TradeStation, Polygon.io & Alpaca for 1-min US stock data via REST API. Analyzing price accuracy, volume & cost, we find surprising

However, when I tested over 15 years of history across 40 futures markets, I found something different.

Peter

consistency and a clear winner for affordable, quality historical data. Essential for algorithmic traders.

Peter

Trading Boosters · Feb 4, 2025

US Stock Momentum Trading System for Retail Traders [Deep research]

I recently tested ChatGPT Pro's Deep Research functionality (released on Monday, February 3, 2025)—currently priced at $200/month—

Peter

CrackingMarkets

Learn to build and run a stable, automated trading framework today.